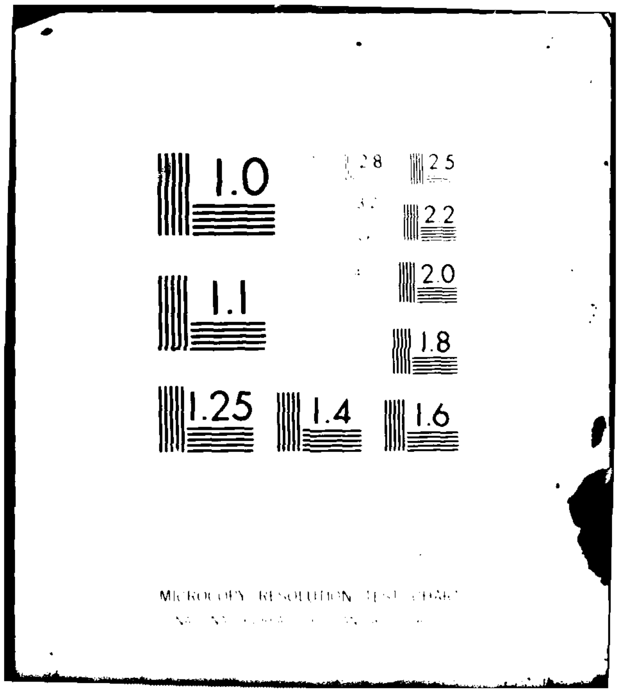

1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
MICROCOPY RESOLUTION TEST CHART

LEVEL II

(12)

AD A111 0383

RADC-TR-81-262
Final Technical Report
October 1981

MICROCODE DOCUMENTATION STANDARDS

Martin Marietta Corporation

J.W. Mathews, Jr.

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

395780

82 02 02 050

RADC-TR-81-262 has been reviewed and is approved for publication.

APPROVED: [illegible]

JAMES L. PREVITE
Project Engineer

APPROVED: [illegible]

JOHN M. MARCINIAK, Colonel, USAF
Chief, Information Sciences Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-81-262	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) MICROCODE DOCUMENTATION STANDARDS		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report 9 Sep 79 - 1 Apr 81
		6. PERFORMING ORG. REPORT NUMBER MRC-81-589
7. AUTHOR(s) J.W. Mathews, Jr.		8. CONTRACT OR GRANT NUMBER(s) F30602-79-C-0247
9. PERFORMING ORGANIZATION NAME AND ADDRESS Martin Marietta Corporation PO Box 179 Denver CO 80201		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 64740F 25260501
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (ISCA) Griffiss AFB NY 13441		12. REPORT DATE October 1981
		13. NUMBER OF PAGES 72
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: James L. Previte (ISCA)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Microcode
Microprogramming
Microcode Standards
Firmware Engineering
Documentation Standards

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Microcode is defined as those instructions residing in either read only or writeable memory that have a one-to-one correspondence to basic operations of hardware elements such as ALU's, multiplexers, selectors, registers, etc.

Although the software development process is well understood and appropriate documentation standards exist to monitor the entire life cycle for software, microcode development to date has no such general understanding, nor has a

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Microcode Documentation Standard yet been established with which to appropriately monitor the microcode life cycle.

This report examines the problems associated with the development of microcode and the attendant problem with establishing a general standard for its development. An approach to the development and documentation of microcode is proposed.

TABLE OF CONTENTS

1.0 INTRODUCTION

This report is submitted to the Rome Air Development Center (RADC) at Griffiss Air Force Base, New York, to support requirements set forth in Contract F30602-79-C-0247. This report addresses the Contract Data Requirements List (CDRL) Number A002 of the contract to which this effort supported; i.e., study of Microcode Documentation Techniques.

Although the software development process is well understood and appropriate documentation standards exist to monitor the entire life cycle for software, microcode development to date has no such general understanding, nor has a Microcode Documentation Standard yet been established with which to appropriately monitor the microcode life cycle. This report explains the problems associated with the development of microcode and the attendant problem with establishing a general standard for its development. An approach to the development and documentation of microcode is proposed.

Microcode is defined as those instructions residing in either Read-Only or Writeable Memory that have a one-to-one correspondence to basic operations of hardware elements such as ALU's, multiplexers, selectors, registers, etc. This definition does not include the programming or instruction codes for a microprocessor. The "micro" portion of microprocessor simply implies that the processor is extremely small (usually on a single chip) and does not mean that the processor is "microprogrammed". Microprogramming and machine language programming are similar in many ways. Both consist of instructions which control the machine processes and serve to solve a particular problem or perform a particular function.

Microprogramming has, however, a number of traits which distinguish it from normal machine language programming.

a. Microcode controls hardware resources at a logically primitive level. For example, microcode might control the input or output of a shifter, an ALU, a status register, a register file, etc. A machine language instruction characteristically handles user defined data and may result in the execution of many microinstructions.

b. Microinstructions may, depending on the machine architecture, control many operations in parallel. These operations are known as micro-operations. Machine language instructions are almost exclusively sequential in nature.

c. Microinstructions are usually stored in an extremely fast memory. High speed memory is expensive and therefore, limited in its use. Machine language instructions usually reside in slower memory or in bulk memory such as disk or magnetic tape.

d. Microinstruction storage (control store) memory tends to be of a wider word size than its machine language main memory counterpart. Typical 16 bit microprogrammed processors, for example, may have from 21 to 56 bit control store word widths.

1.1 HISTORY

Microprogramming as an approach to digital control was first introduced in the 1950's by Professor Maurice V. Wilkes of Cambridge University Mathematical Laboratory. The concept was seldom used until International Business Machines (IBM) introduced their System 360 in the mid 1960's. Wilkes' approach was described in a 1951 paper [1]. His purpose was to propose a method for systematic, less complex control of a "calculating machine".

Early computers contained a control element which generated control signals to route and synchrcnize data between registers, buses and logical devices such as an ALU. Wilkes' microprogrammed approach allowed the control signals to be retained in a control memory rather than developed by a complicated control element.

At the time Wilkes proposed his "new technique", vacuum tubes and relays formed the switching and logic devices. Memory which would have stored Wilkes' "micro-operations" would have added significant delays to the already slow (by today's standards) logic to produce yet slower machines.

It was the availability of significantly faster memories that enabled microprogramming to become a viable alternative to the pure logic approach. When IBM introduced their 360 series in 1964, they microprogrammed all but the fastest and most complex machines. One of the main reasons for these mainframes to be microprogrammed was to provide emulation of previous IBM processors, thus allowing customers to retain and use their existing software. Software "compatability" has continued to be a significant driving force in processor design today.

The initial vehicle for microprogram storage was Read Only Memory (ROM). The instructions in the ROM were called microinstructions and the ROM storage was called control store. With the advent of high speed random access read/write memory, the concept of "writeable" control store was developed. Processors which were black boxes, executing only manufacturer defined instructions, now could be made to execute peculiar user defined instructions or routines.

1.2 UNDERSTANDING MICROCODE

Initially, microcode was used only to simplify or generalize the implementation of machine language computer instruction sets. With the advent of bit slice chips and microprogrammable architecture in smaller and smaller high performance packages, microprogrammed devices have become the norm where hard-wired logic systems were once the standard.

The goal of this study is to aid in the development of specifications and guidelines to ensure that microcode delivered to the Air Force is of good quality, understandable and maintainable.

For microcode, maintenance includes finding and correcting errors as well as possibly modifying or enhancing the microcoded device's capabilities. In some cases, microcode also lends itself to perform built-in test functions so that hardware errors may be diagnosed. None of this is possible, however, unless the hardware technician can be given documentation adequate to understand the hardware/microcode system to be maintained.

There are both advantages and disadvantages when comparing microprogrammed logic with hardwired logic.

a. Microprogrammed logic tends to be less expensive than hardwired logic in terms of its development or implementation costs. This, of course, assumes that both approaches are viable alternatives to the design problem.

b. Microprogrammed logic is more flexible than hardwired logic. In terms of microprogrammed computers, new instructions as well as entire algorithms can be added usually without modification (ROM chips excepted). For devices other than CPU's, new or modified capabilities can usually be added by changing the microcode and installing new ROM chips.

c. Microprogrammed logic is usually simpler to modify, either to correct an existing error or to change or enhance an existing function.

d. Microprogrammed logic is usually slower than the equivalent hardwired logic. Fortunately, not all design problems require hardwired "state-of-the-art" speed. Where speed is a concern, a modified microcoded design can sometimes enhance the overall speed to meet device or system requirements.

In order to better understand the varying requirements for design procedures, development t^chniques and documentation, it is necessary to review the differing mi[illegible] code formats and their use within machines or devices.

1.3 MICROCODE TYPES

Microcode instruction types are usually characterized as either vertical or horizontal. In present day machines, microinstructions are typically combinations of both, known as diagonal or hybrid instructions.

Vertical microinstructions are very much like normal machine language instructions in that they typically have an operation code and two or three arguments or operands. Word size is relatively narrow (12 to 24 bits) in comparison to a horizontally structured microcode instruction (60 to 80 bits). Hypothetical vertical and horizontal microinstruction formats are shown in Figures 1 and 2 respectively. Other characteristics of vertical microinstructions include:

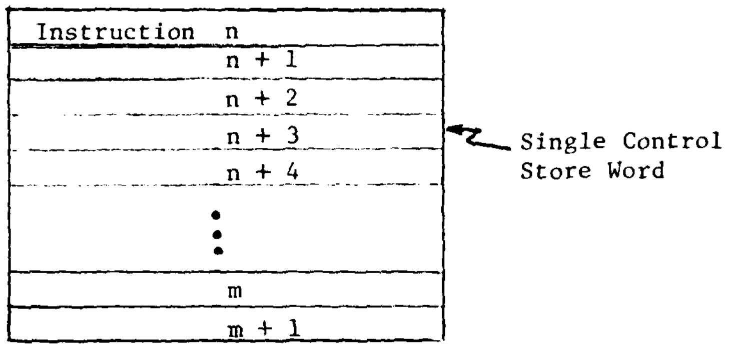
a. Most, if not all, bit combinations are defined.

b. Execution time per instruction is usually longer because several clock cycles may be needed to complete one instruction cycle.

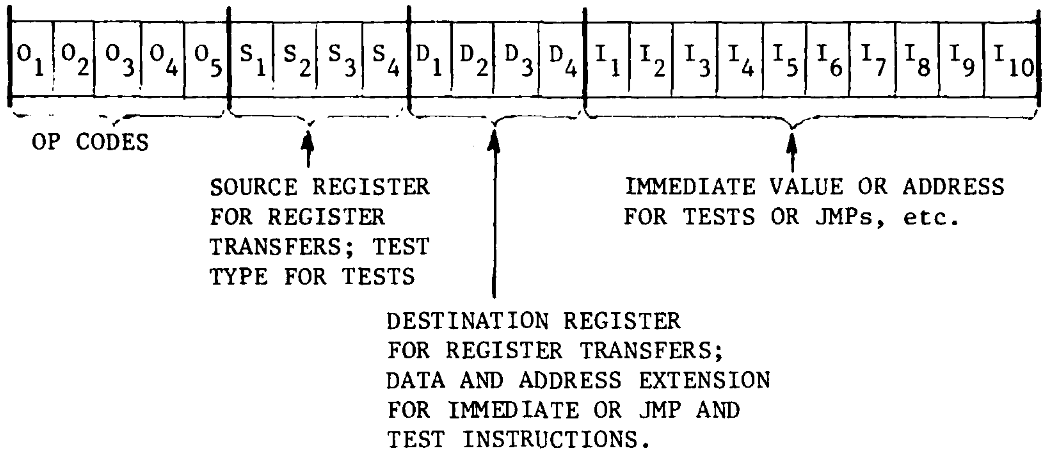
c. Fields are usually decoded, i.e., 4 bits select one of 16 possibilities rather than each bit representing one of four independent options.

The organization of the control store containing microinstructions can be an additional influence on the complexity of development and documentation of microcode. The following examples illustrate only two of the many possible organizations.

Example 1.



Example 1 shows the simplest of organizations. It is one microinstruction per control store word. The microinstruction may be either vertical, horizontal or a diagonal combination.

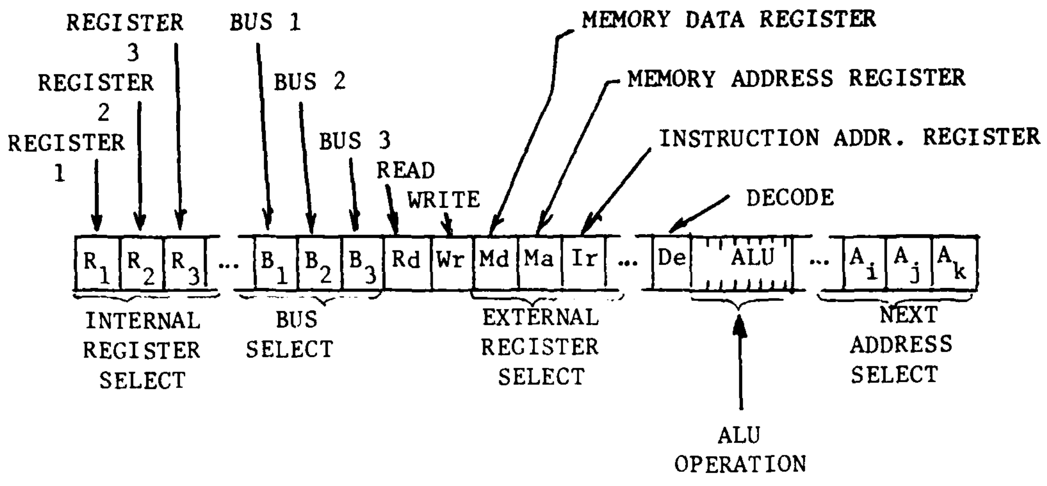

O_1 O_2 O_3 O_4 O_5 S_1 S_2 S_3 S_4 D_1 D_2 D_3 D_4 I_1 I_2 I_3 I_4 I_5 I_6 I_7 I_8 I_9 I_{10}
OP CODES
SOURCE REGISTER FOR REGISTER TRANSFERS; TEST TYPE FOR TESTS
IMMEDIATE VALUE OR ADDRESS FOR TESTS OR JMPs, etc.
DESTINATION REGISTER FOR REGISTER TRANSFERS; DATA AND ADDRESS EXTENSION FOR IMMEDIATE OR JMP AND TEST INSTRUCTIONS.

OP CODES: ADD, ADD IMMEDIATE, XOR, XOR IMMEDIATE, SUB, SUB IMMEDIATE, AND, AND IMMEDIATE, OR, OR IMMEDIATE, LOAD IMMEDIATE, MOVE, DECODE, etc.

REGISTERS: GENERAL REGISTERS: $R_0, R_1, \ldots, R_7$;
INTERFACE REGISTERS: MEMORY ADDR. REGISTER (MAR),
OPERAND ADDR. REGISTER (OAR).
SPECIAL REGISTERS: STATUS, INTERRUPT, etc.

Note - There is a definite similarity to "normal" computer instructions. Each field is decoded to select a unique value.

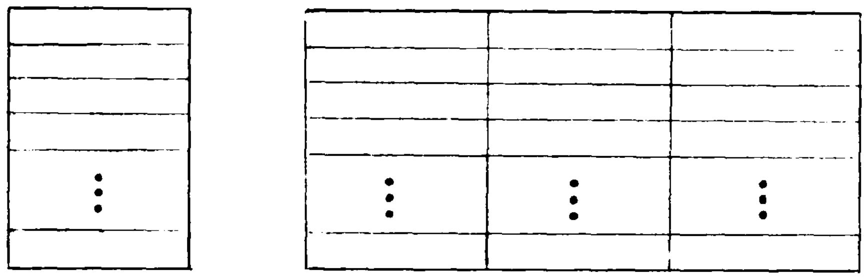
Figure 1 - A Hypothetical Vertical Microinstruction Format.



Note - Most fields are not decoded thus allowing multiple functions and data paths. It is possible to select combinations that are not useful.

Figure 2 - A Hypothetical Horizontal Microinstruction Format.

Example 2.



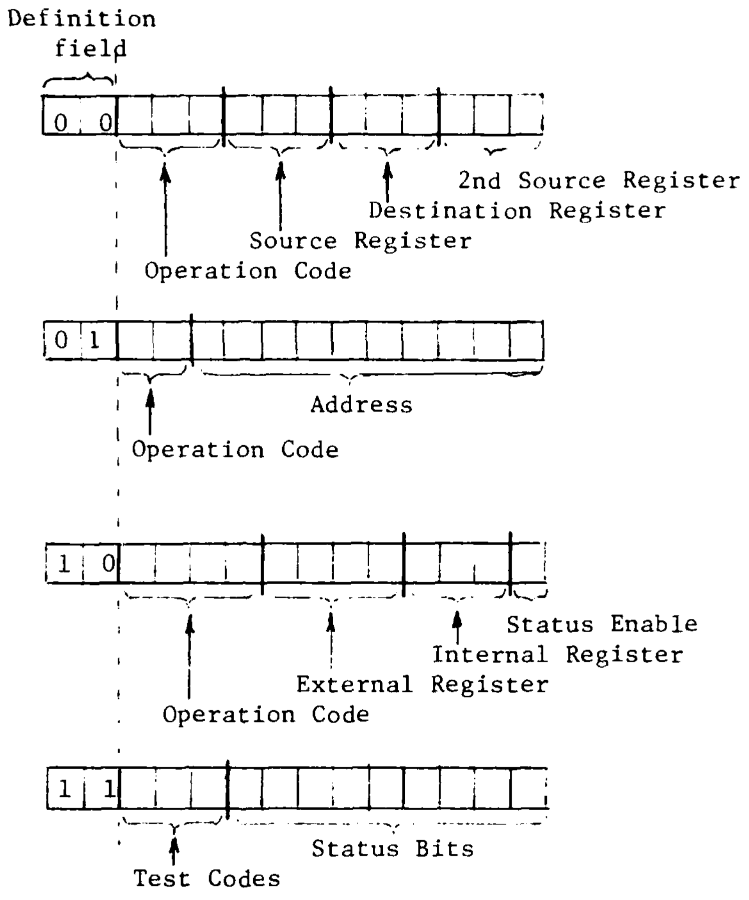
(Part 1)
Single Instructions
Per Control Store Word

(Part 2)
Multiple Instructions
Per Control Store Word

Example 2 shows a more complicated organization in which there are two parts to the control store. The first part is a single microinstruction per control store word. The second contains multiple instructions per word, called nanoinstructions. The nanoinstructions execute the microinstructions in the same fashion that microinstructions execute computer machine language instructions: A single instruction is fetched from the "Part 1" control store. It is then decoded and executed by the nanoinstructions in the "Part 2" control store. Multiple nanoinstructions can be fetched in a single access. An architecture similar to that of example 2 has been used in a machine which emulates other machines.

As was previously stated, today's machines are usually not purely horizontal or purely vertical but a combination of both or diagonal. It is wasteful to allow single bits for each possible function if they represent, in fact, mutually exclusive operations. An ALU, for example, can usually only perform one operation (add, subtract, etc.) at one time, so it can shorten the microcode instruction word width if mutually exclusive operations or resources are encoded into fields. Such encoding does not make a microinstruction less "horizontal." In the interest of economy, it is also possible to encode "almost" exclusive resources. The rationale is that the need to control these resources simultaneously is almost nonexistent, therefore speed and/or absolute flexibility can be slightly comprised. The diagonal microinstruction has thus been justified.

Another method of shortening the control word is to allow one of the fields within the microinstruction to define the encoding of the other fields in the microinstruction. Thus, under different field definitions, a microinstruction might have the following hypothetical formats.



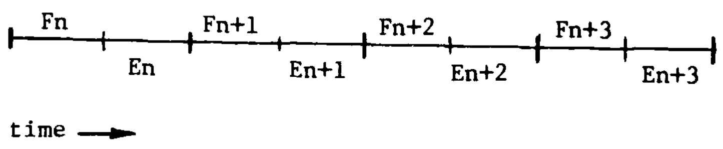
Example 3.

This encoding can significantly shorten the required width of the control store, although extra time and interpretation logic are necessary to execute the instructions. The understanding of the microcode as well as the development of the microcode for such a device can be formidable.

In addition to microinstruction format and control store organization as factors in microcode complexity, timing within the instruction execution is a significant factor. In a group of microinstructions, each perform the fetch and execute sequence associated with the machine language instructions of a computer. Timing can be considered in two areas: intra-instruction and interinstruction timing.

Interinstruction timing may take two forms: serial and overlapped. The serial form is simply end-to-end fetch and execute phases of each microinstruction in sequence.

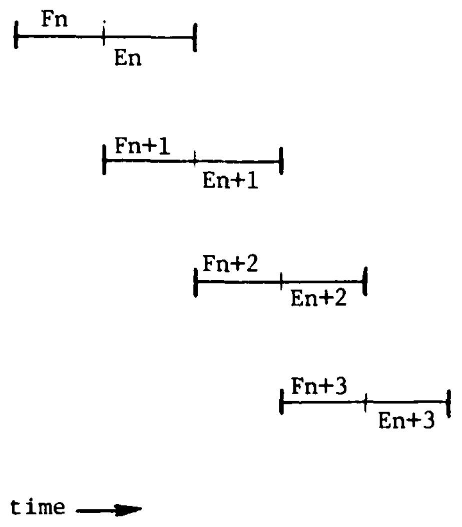
Example 4.



In the above diagram, Fn represents the instruction fetch portion of the nth microinstruction while En represents the execution portion.

The overlapped form of timing performs portions of successive instructions in parallel.

Example 5.



The above diagram shows completely overlapped fetch and execute portions of successive instructions. Although this technique is obviously faster than the serial method, it adds a level of hardware complexity and microcode complexity as well. Consider for example, the execution of a "JUMP" or a "SKIP" microinstruction during period n. If the next control store address (CSA) is calculated during En then En+1 must be delayed one fetch cycle or at least one or more minor clock cycles.

An alternative approach is to calculate the next CSA during the fetch cycle. This puts an added burden on the intra-instruction timing of the execute cycle to insure that testable conditions for conditional skips or jumps are available and are valid during the parallel fetch cycle. In overlapped instructions then, intra-instruction timing may be critical. Microcoded designs are not limited to two overlapped instructions; multiple overlapped instruction designs have been implemented.

Examples of items or events which may affect or be affected by intra-instruction timing are:

a. Perform skip test;
b. Generate next CSA;
c. Initiate interrupts;
d. Generate Control Word;
e. Initiate Fetch;
f. Load control word register;
g. Load CSA register;
h. Set or reset special conditions;
i. Perform micro stack operations;
j. Decrement micro loop register;
k. Select source bus data;
l. Execute ALU/Shift operations;
m. Generate dynamic condition codes;
n. Select destination bus;
o. Store destination data;
p. Update condition codes;
q. Initiate execute mode.

These examples were taken from an actual microcoded processor designed to emulate other processors. The list is by no means complete, however, it serves to indicate a level of complexity that may be reflected in the microcode required to efficiently control such a processor.

For many years the vast majority of microcoded devices were large and medium scale computers. With the advent of bit slice architectures in which bit slice devices can be combined to form a spectrum from the simplest to the most complex of systems, microprogrammed devices, controllers, recognition devices and high speed devices which can take the place of computers.

In the typical 16 bit minicomputer for example, registers, databases, the ALU, etc., are designed to facilitate control and efficiency in dealing with 16 bits of data. Bit slice processors perform similar functions for 2 to 4 bits. These processors may be combined to implement wider registers, ALU's, etc. If the end result is to be a computer, the bit slice devices may be combined with memory, sequencers, decoders, interrupt arbitrators, display drivers, etc.

2.0 THE SURVEY

In an effort to research the differing aspects of microcode generation, the study examined the problem from three viewpoints:

a. Search existing published literature.

b. Interview users and developers of microcoded devices.

c. Perform a more detailed analysis of our own microcode and software development.

It was felt that published literature might reveal new or future developments through research or university studies whereas interviews with users would reveal the actual techniques in use today.

2.1 PUBLISHED LITERATURE

The search for pertinent published literature began with the use of library search services. The "DIALOG" Information Retrieval Service (Lockheed Information System) was accessed and the following files were searched:

a. COMPREHENSIVE DISSERTATION ABSTRACTS. (File 35) This file is a guide to American dissertations by subject, title and author at accredited institutions. Some Canadian and overseas papers are also included. There are approximately 650,000 entries.

b. CONFERENCE PAPERS INDEX. (File 77) Subject areas contained in this file include life sciences, chemistry, physical sciences, geosciences and engineering. This file is a source of information on reports of research and development from papers presented at conferences and meetings. There are more than 700,000 entries.

c. INSPEC. (File 12 or File 13) This file forms a database for material contained in journals in the fields of physics, electrotechnology, computers and control. There are actually two files where File 12 contains data from 1969 through 1977 and File 13 contains data from 1978 to the present. File 13 was utilized for this study. Both files have a total of over 1,000,000 entries.

d. SMITHSONIAN SCIENCE INFORMATION EXCHANGE CURRENT RESEARCH. (File 65) This file is a database of government and privately funded scientific research projects that are in progress or were initiated and completed during the last two years. There are approximately 300,000 entries.

e. NATIONAL TECHNICAL INFORMATION SERVICE: NTIS. (File 6) This database contains listings of government sponsored research, development and engineering plus analysis. There are over 750,000 entries.

f. MAGAZINE INDEX. (File 47) This database contains over 300,000 entries from more than 370 popular magazines with varied subjects including science and technology.

Each of the above files may be accessed by the use of keywords. For example an access keyword of "MICROCODE?" would allow any file entries dealing with microcode or microcoding to be listed. Keywords were grouped into three sets:

Set 1)	MICROCOD?
	MICROPROGRAM?
	MICROINSTRUCTION?
	MICROOPERATION?
Set 2)	DEVELOPMENT?
	DOCUMENT?
	STANDARD?
	CODING
	TESTING
Set 3)	TOOL?
	AID?
	COMPILER?
	COMPILING PROGRAM?
	TECHNIQUE?
	METHODOLOG?

In order to trim the total number of publications, the elements of each set were "ORed" together then set 1 was "ANDed" with Set 2 and Set 3. Thus, any subject concerning MICROCODE or MICROCODING or MICROINSTRUCTIONS, etc. and DEVELOPMENT or DOCUMENTS or DOCUMENTATION, etc. would be extracted. The files with these attributes produced the following number of possibly useful articles and papers:

File / Keyword	35	27	13	65	6	47
Set 1 and Set 2	1	5	174	16	65	0
Set 1 and Set 3	2	13	179	18	64	5

An additional search was implemented using the NASA RECON (Remote Console) database. The file searched included International Aerospace Abstracts, NASA Tech Briefs, as well as other files. The subjects selected were:

a. MICROPROGRAMMING;
b. DEVELOPMENT;
c. DOCUMENTATION;
d. STANDARDS;
e. STANDARDIZATION;
f. TOOLS;
g. METHODOLOGY;
h. COMPILERS.

Subjects 2 through 8 were then "ORed" together and "ANDed" with subject 1 to form the keyword scan. Thus, subjects having to do with MICROPROGRAMMING DEVELOPMENT or MICROPROGRAMMING DOCUMENTATION, etc. would be listed. The search resulted in the tabulation of 28 possible pertinent documents. Another search utilized the resources of the Defense Technical Information Center (DTIC) under the Department of Defense. Primary or first level search terms included:

a. MICROCODE (and its derivatives);
b. MICROCOMPUTERS;
c. MICROPROCESSORS.

Secondary or second level search terms included:

a. ASSEMBLY LANGUAGES;
b. AUTOMATIC PROGRAMMING;
c. COMPILERS;
d. COMPUTER PROGRAM DOCUMENTATION;
e. COMPUTER PROGRAM RELIABILITY;
f. COMPUTER PROGRAM VERIFICATION;
g. COMPUTER PROGRAMMING;
h. COMPUTER PROGRAMS;
i. COMPUTER SEQUENCES;
j. DEBUGGING (Computers);
k. EXECUTIVE ROUTINES;
l. FIELDS (COMPUTER PROGRAMS);
m. FORTRAN;
n. GOAL PROGRAMMING;
o. HIGH LEVEL LANGUAGES;
p. MACHINE CODING;
q. MICROPROGRAMMING;
r. MICROCODE (and its derivatives);
s. NON CONVEX PROGRAMMING;
t. PROGRAMMING LANGUAGES;
u. SIMULATION LANGUAGES.

This search resulted in the listing of approximately 1,100 possible pertinent documents.

In addition to automated searches, the past five years of "Reader's Guides" were examined and local area technical libraries were visited to find appropriate relevant material.

In order to further reduce the number of publications to a reasonable quantity, abstracts of the articles listed by the automated searches were examined. Classified and foreign language publications were eliminated, as well as those with doubtful technical content. Even with this exclusion process, the automated searches, Reader's Guide references and library visits yielded more than 100 articles and publications which were examined during the course of this study.

2.2 CORPORATE SURVEY

In order to survey the tools, techniques, and procedures as utilized by industry today, a series of tasks were performed.

a. Select a candidate list of corporations which were possible users or developers of microcode or microcoded devices.

b. Try to locate a cognizant person or persons within that organization.

c. Develop a set of relevant but comprehensive questions,

d. Design a convenient, easy to fill out questionnaire utilizing those questions.

The questionnaire was circulated to the cognizant personnel at selected corporations. The result of this effort is tabulated in Section 2.2.1 below.

2.2.1 CORPORATE SURVEY TABULATION

There were approximately 200 questionnaires sent out to corporations which used or produced microcoded devices. The response cannot be considered successful from the standpoint of the numbers of respondents. Approximately 20 returned the questionnaires with positive responses (that they were users or producers). A somewhat greater number (approximately 30) responded negatively and the remainder chose not to respond in any fashion. The results can be considered successful however, in that there were some clear cut trends by those that responded positively. These responses are cataloged briefly in the following paragraphs. It should be noted that not all respondents answered all questions.

2.2.1.1 USERS OF MICROCODED DEVICES

	Yes	No
a. Those that had failures of the devices	11	1
b. Those that:		
1. Discarded the failed devices	9	0
2. Returned them to manufacturer	4	5
3. Tried to repair devices	2	6
Repair Successful?	2	0
Documentation Adequate?	2	0
c. Modified or enhanced a microcoded device?	3	8
Documentation Adequate?	2	3

2.2.1.2 DEVELOPERS OF MICROCODE DEVICES

The results of the Section of the questionnaire relating to developers of microcoded devices is shown below.

As in the user poll, not all questions were answered. It should also be noted that questions relating to willingness to discuss problem areas or data representation have been omitted.

	Yes	No
a. Was a planning phase executed?	18	0
If so, was documentation produced?	17	1
Was there a predefined standard?	4	13
Were software or hardware tools or aides used?	6	11
b. Was a requirements phase executed?	16	0
If so, was documentation produced?	15	2
Was there a predefined standard?	8	9
Were software or hardware tools or aides used?	4	11
c. Was a hardware design phase executed?	18	0
If so, was documentation produced?	18	0
Was there a predefined standard?	13	5
Were software or hardware tools or aides used?	14	4
d. Was a microcode design phase executed?	18	0
If so, was documentation produced?	18	0
Was there a predefined standard?	9	9
Were software or hardware tools or aides used?	14	4

		Yes	No
e.	Was a microcode and checkout phase executed?	17	0
	If so, was documentation produced?	15	2
	Was there a predefined standard?	9	6
	Were software or hardware tools or aides used?	14	2
f.	Was a simulation of the microcode performed?	10	6
	If so, was documentation produced?	6	8
	Was there a predefined standard?	5	5
	Were software or hardware tools or aides used?	10	3
g.	Was there a translation to PROM/ EPROM, etc. format?	17	1
	If so, was documentation produced?	14	4
	Was there a predefined standard?	8	6
	Were software or hardware tools or aides used?	14	1
h.	Was functional testing performed?	17	0
	If so, was documentation produced?	8	8
	Was there a predefined standard?	6	2
	Were software or hardware tools or aides used?	14	2
i.	Was functional testing performed?	14	3
	If so, was documentation produced?	8	6
	Was there a predefined standard?	7	1
	Were software or hardware tools or aides used?	10	4
j.	Was system text and validation performed?	17	1
	If so, was documentation produced?	11	4
	Was there a predefined standard?	10	1
	Were software or hardware tools or aides used?	13	1
k.	Was an installation and maintenance phase executed?	13	3
	If so, was documentation produced?	12	0
	Was there a predefined standard?	10	2
	Were software or hardware tools or aides used?	9	4

		Yes	No
l.	Was a training phase executed?	11	5
	If so, was documentation produced?	9	2
	Was there a predefined standard?	8	1
	Were software or hardware tools or aides used?	8	3
m.	Were requirements separated into microcode and hardware requirements?	10	8
n.	Was an assembler used in code and checkout?	15	3
p.	Any higher order language tools?	7	10
q.	Did documentation for users or maintenence personnel cover:		
	Critical timing information?	11	6
	Sequencing information?	15	2
	Operation information?	15	1
	Instruction and/or data representations?	16	1
	External interface information?	16	1
	Internal interface information?	15	1
	Mode information?	11	5
r.	Separate documentation for microcode?	15	2
s.	Does your company intend to develop aids for developing microcode and/or documentation?	14	3
t.	Maintenance personnel expected or required to have detailed architectural or timing knowledge?	5	10
u.	Do your customers give the option of modifying microcode?	4	12
v.	Effort made to optimize microcode (time or space)?	16	1
w.	Product have selfcheck or fault recovery features in microcode?	10	5

No effort will be made to analyze responses to each specific question. General survey results together with information gathered from subsequent interviews, and literature searches will be presented in following sections.

2.3 STUDY RESULTS

Examination of the questionnaire tabulations indicates that most developers of microcoded devices do indeed go through the development phases of planning, requirements, hardware design, microcode designs, code and checkout, functional test, and system test and evaluation. Subsequent interviews with these developers revealed that none actually went through all phases in a distinct, sequential fashion. Comparison of the individual responses indicated that microcode development fits more easily into five phases. These five phases are Analysis, Design, Code and Checkout, Test and Integration, and Operations and Maintenance. These five phases are described in detail in Section 3.0 of this report.

One unexpected finding of the study involved the subject of "predefined standards". Although a number of respondents stated that predefined standards were followed, for certain phases of the microcode development process, further interviews revealed that none were "company-wide" standards specifically addressing microcode. Standards (if any) tended to be project unique and informal. Some developed their microcode using software standards.

Another finding pertinent to this project was that in the majority of cases, separate documentation was produced for microcode. Further investigation revealed that this did not necessarily mean that separate "stand alone" documentation was produced for microcode but that a separate section in either a hardware document or a software document (or both) may have been established. It was admitted that for some project where a microcoded device was utilized in place of hardwired logic (i.e. a sequencer) it was hidden in the other hardware documentation. This was done so that the microcoded device wouldn't have to be documented to software standards.

2.3.1 PROBLEM AREAS

In discussing the problems encountered in the microcode development process with producers and developers of microcode, it became obvious that, except for difficulties such as parts availability or access to development tools, the problems tend to fall into four categories.

a. Incorrect, misunderstood, incomplete, or changing requirements to be satisfied by the microcode or the microcoded device.

b. Incorrect, misunderstood, or incomplete specifications of the hardware features and characteristics of the microprogrammable device or its interfacing components.

c. Confusion relating to the microcode instruction set itself. This was usually due to a complex instruction format and/or an overlapped execution sequence with complex timing considerations.

d. Difficulty with respect to microcode check out and component or system integration. Microcode is inherently more difficult to check out and debug since most microcoded devices do not provide convenient operator interface capabilities. In some instances, logic analyzers or other hardware devices are the only means of monitoring microcode execution.

The problems described in items a, b, and c above can be minimized through proper planning procedures, and documentation. The problem cited in item d can be alleviated through the use of design tools such as software simulators in conjunction with debug and trace routines. Documentation and development tools are described in greater detail in the following sections.

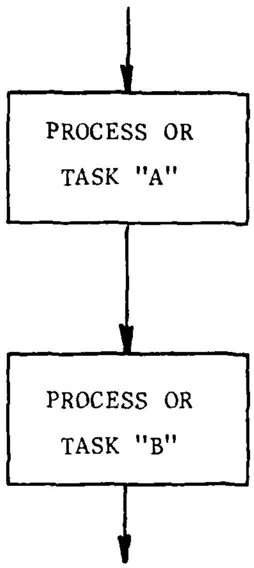
2.3.2 MICROCODE DEVELOPMENT TOOLS

Survey results indicate that software tools to aid in the planning and requirements allocation (analysis phase) portions of software development are compatible with the microcode development process. Where the compatibility of tools begins to break down is in the design, code and checkout phases of development.

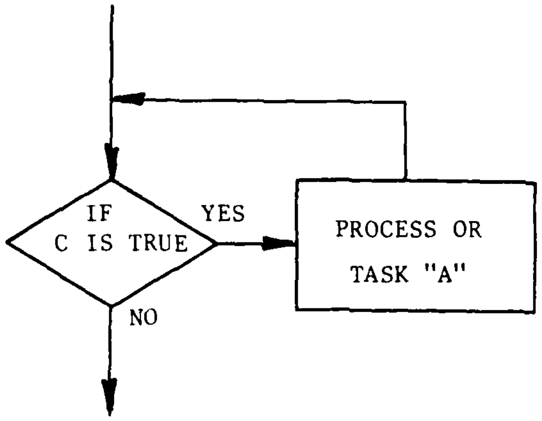
2.3.2.1 DESIGN TOOLS AND TECHNIQUES

Although there is research being done in the area of automated microprogramming design, the majority of producers use a manual top-down structured design approach originated for software. Functional breakdown to levels of greater and greater detail appears to be the most popular approach. As the detail designs approach their final stages, differences appear in the design representations.

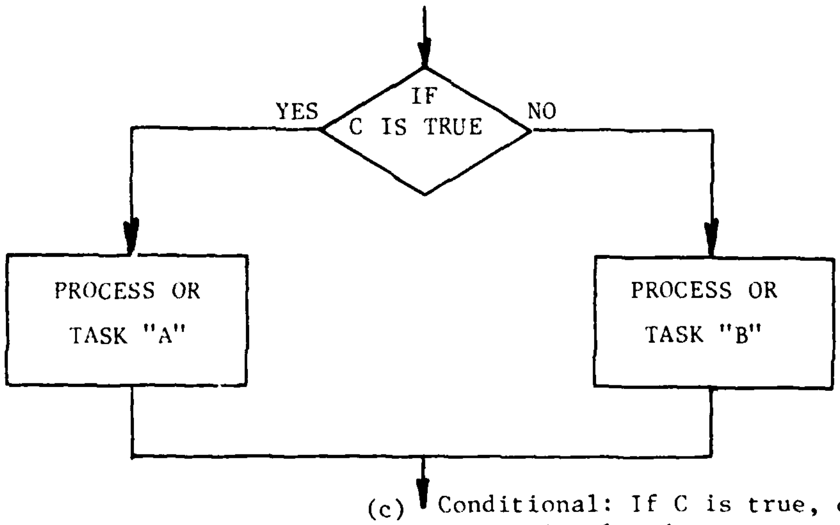
One of the most effective means of representing a microcode design is through the use of visual diagrams. There are many types of diagrams, however the two most commonly used categories of diagrams are the flow (see Figure 3) and the microprogram diagram or "microblock" (see Figures 4a and 4b).



(a) *Sequential: Process* A followed by process B.



(b) Do While: While C is true, do process A.



(c) Conditional: If C is true, do process A, else do process B.

Figure 3 - Typical Microprogram Flow Chart Structures

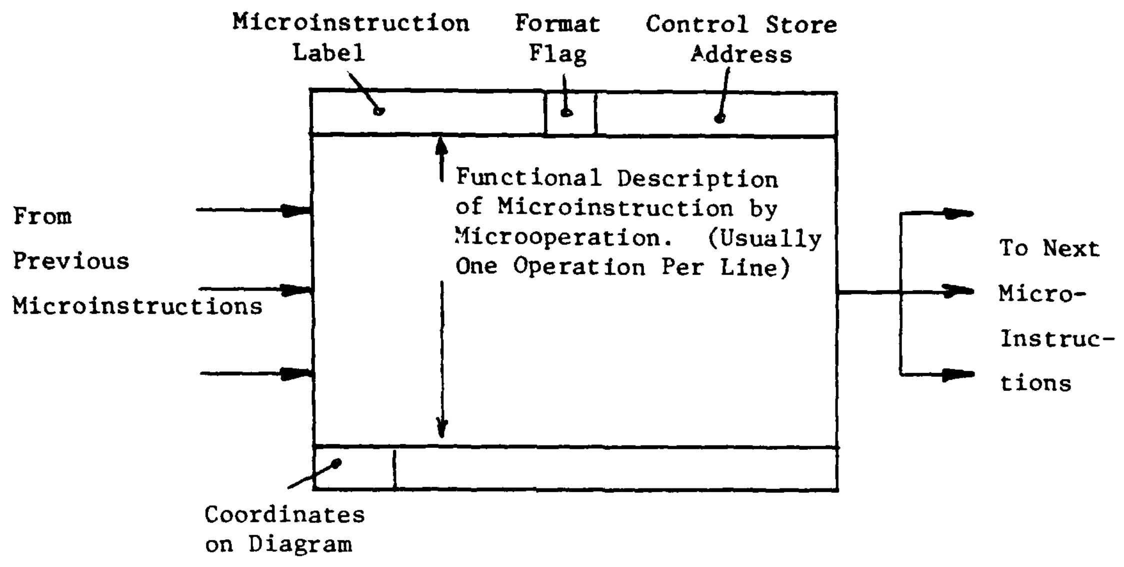

Microinstruction Label
Format Flag
Control Store Address
Functional Description of Microinstruction by Microoperation. (Usually One Operation Per Line)
From Previous Microinstructions
To Next Micro-Instructions
Coordinates on Diagram

Figure 4a - Microinstruction Block Produced under "MIKADO" [6]

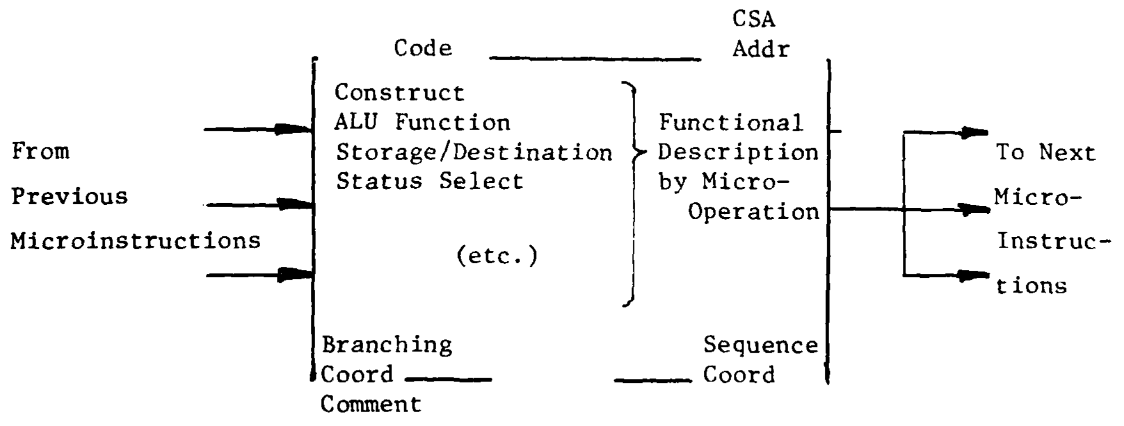


Figure 4b - Microinstruction Block Similar to that Used to Describe the System/360 Model 30.

The notation used in flow charts ranged from very simple to the very difficult in which the reader would require a thorough understanding of both the device characteristics and the symbolic representations for registers, buses, etc. Figure 5a is an example of a relatively simple flow chart. Even though device specific notation was used this type of flow chart can be made much more understandable if comments and microstore address information is added to the diagram (see Figure 5b). The flow chart shown in Figure 5b is not convenient for more horizontally organized instruction sets in which multiple activities or functions may be concurrent. In these cases, users have tended to use a flow chart box or microblock. An example of a rather complicated microblock for a device with diagonal 56 bit micro instructions is shown in Figure 6. It is obvious that familiarity with the device and the microcode format is a necessity if one is to use that particular diagram. These boxes are sequentially strung together to form a "flow chart".

Another example of a flow chart box for a two part instruction (microinstruction and nanoinstruction) is shown in Figures 7a and 7b. Here, all the parallel data transfers are shown in a single sub-block, while control functions and labels are shown in separate attached blocks. The user still must be familiar with the symbolic representation for registers, buses and functions.

These boxes have an advantage in that they can be generated fairly easily on a line printer. This then leads to the possibility of utilizing a terminal with limited graphics capability to aid in the semi-automated generation of functional diagrams. A number of microcode developers state that gaining this capability was one of their future projects. A translation from this format to assembly language input is not a difficult task.

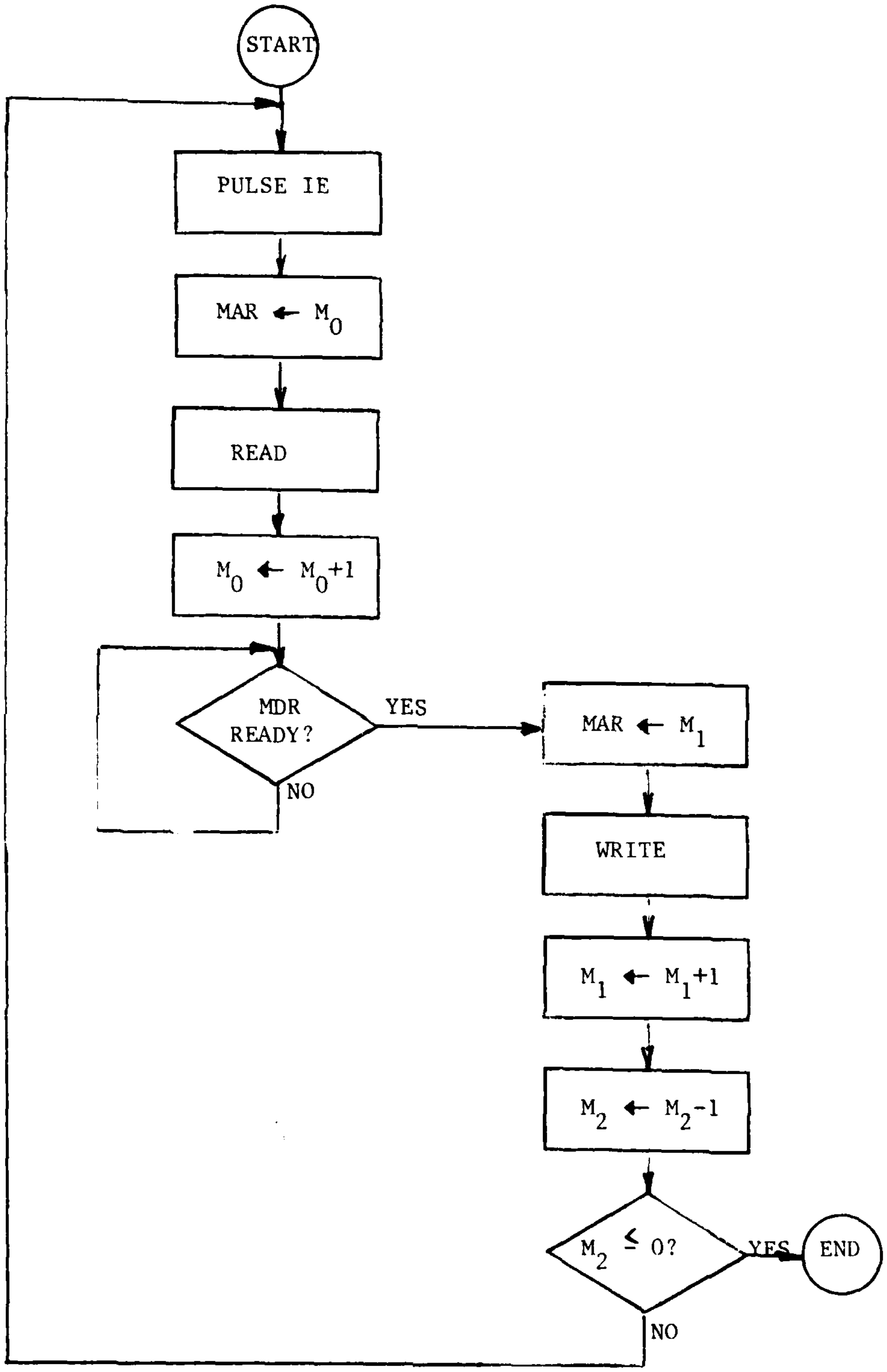


Figure 5a - A Microcode Flow Chart of a "Move Block" Function.

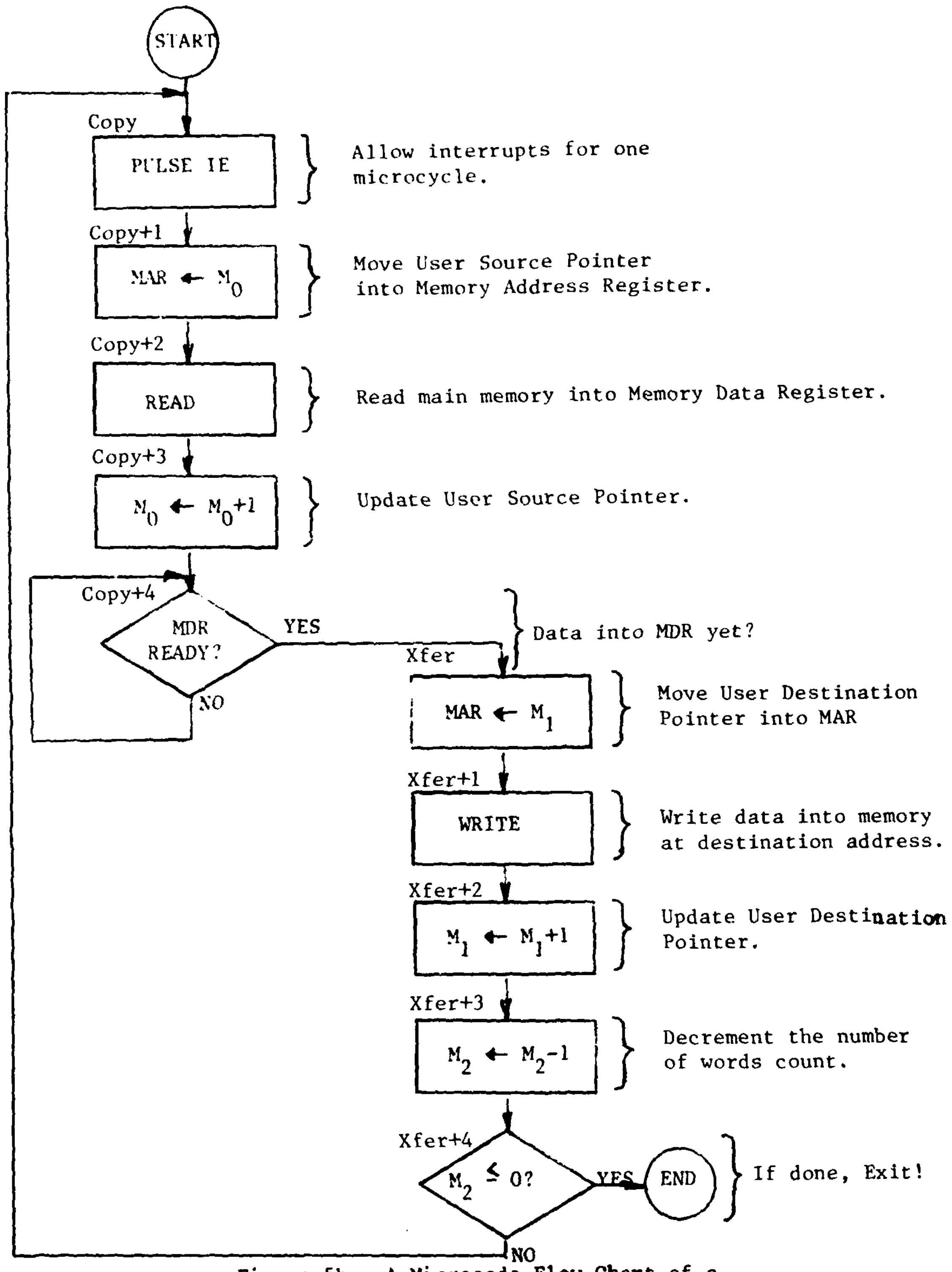

START
Copy
PULSE IE
Allow interrupts for one microcycle.
Copy+1
MAR ← M_0
Move User Source Pointer into Memory Address Register.
Copy+2
READ
Read main memory into Memory Data Register.
Copy+3
$M_0 \leftarrow M_0+1$
Update User Source Pointer.
Copy+4
MDR READY?
YES
NO
Data into MDR yet?
Xfer
MAR ← M_1
Move User Destination Pointer into MAR
Xfer+1
WRITE
Write data into memory at destination address.
Xfer+2
$M_1 \leftarrow M_1+1$
Update User Destination Pointer.
Xfer+3
$M_2 \leftarrow M_2-1$
Decrement the number of words count.
Xfer+4
$M_2 \leq 0$?
YES
END
If done, Exit!
NO

Figure 5b - A Microcode Flow Chart of a "Move Block" Function. (Information Added)

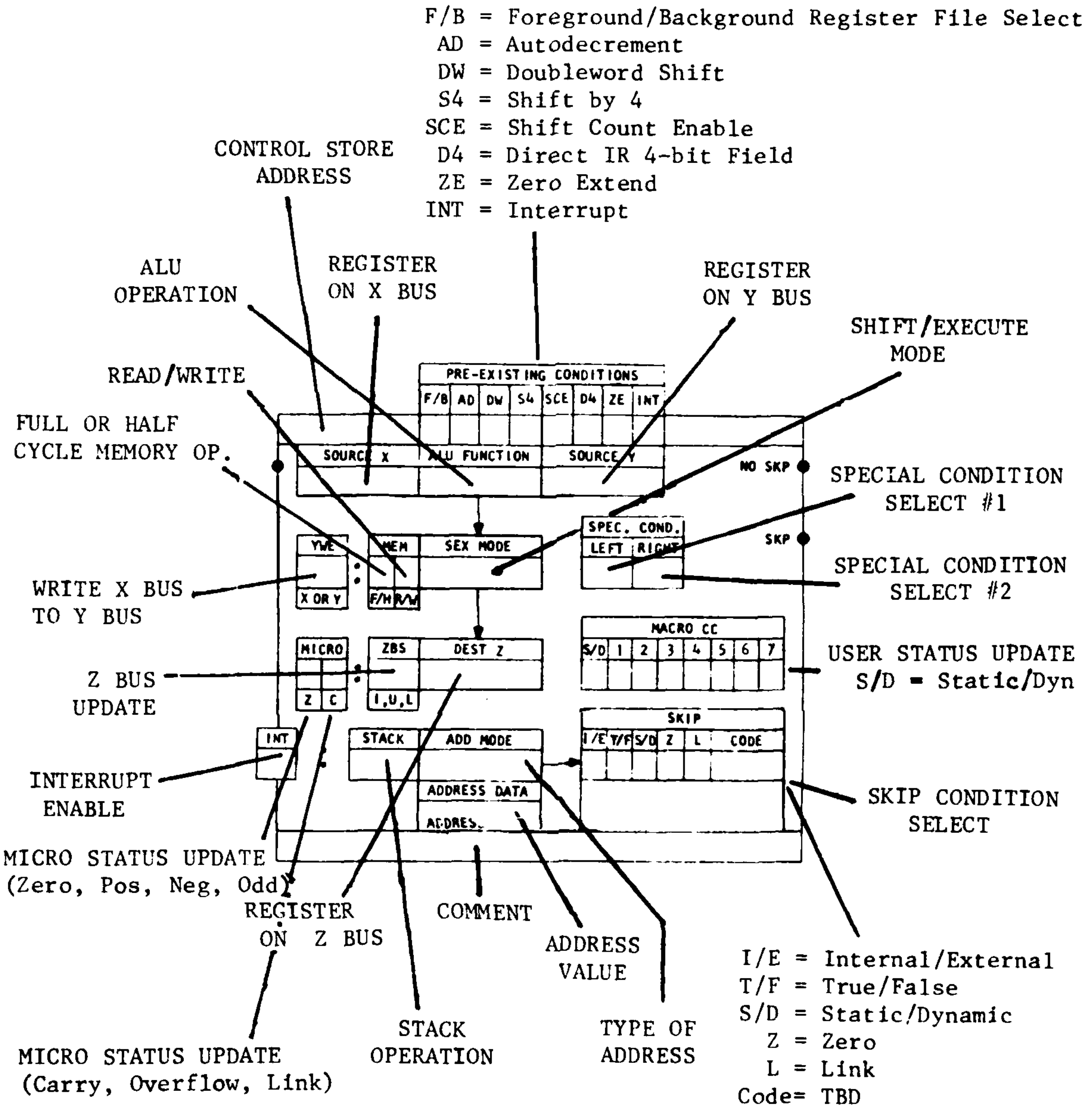


Figure 6 - An Example of a Complicated Control Block

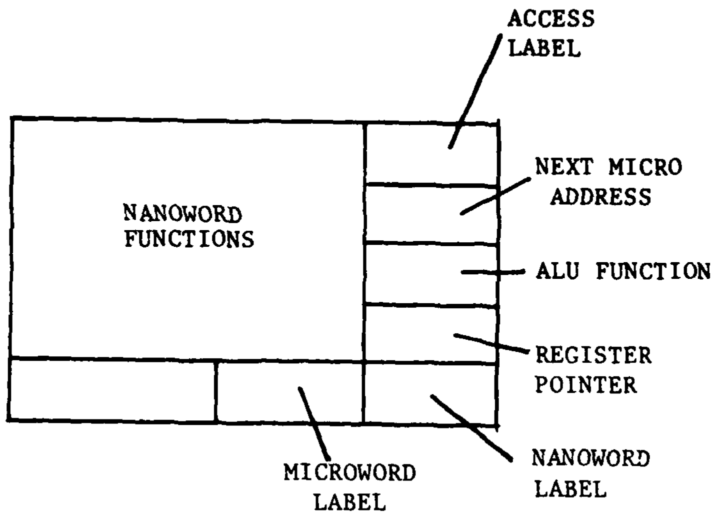


Figure 7a - General Format of a Flow Chart Box [4]

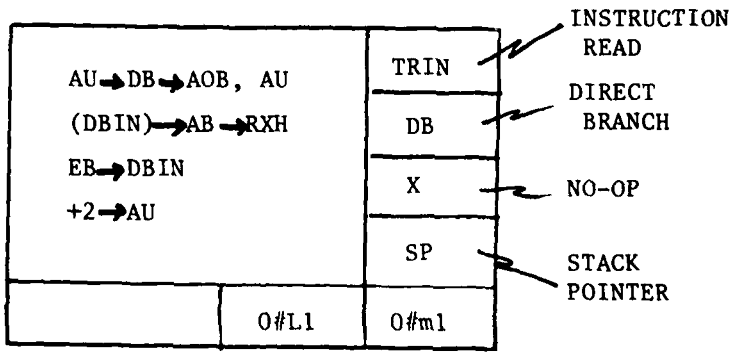
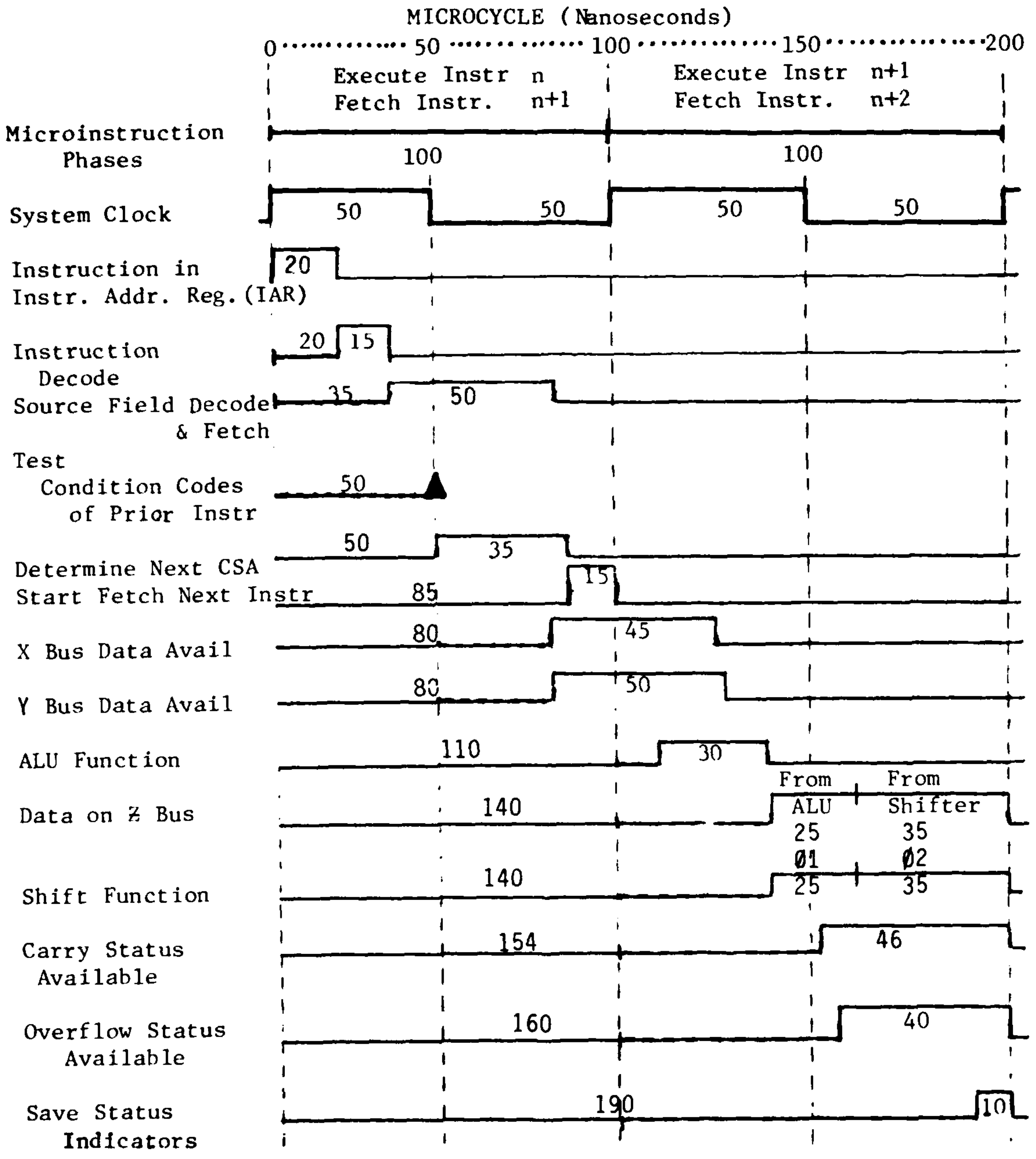


Figure 7b - Example of a Flow Chart Box [4]

One other popular means of describing a microcode design is through the use of a language called "PDL" (Program Design Language). Its advantage in the design phase is that stepwise refinement of functional levels is conveniently accomplished without having to "re-draw" a flow chart. The language uses structured coding constructs such as DO WHILE, CASE, and IF THEN - ELSE. Although the microcoder is very rarely given the luxury of enough executing time or control store to utilize fully structured code, a PDL can be a useful tool to insure a good functional design. Two other design aids should be introduced in this section. The first is a timing diagram. Many of the problems defined during the study stemmed from the lack of sufficient timing information. In some cases severe rework of the detailed design had to be done because either interinstruction or intra-instruction timing was not well defined. Data signals which might be used or tested by microcode should be displayed on a timing diagram. An example is shown in Figure 8. The diagram should reflect worst case signal availability and should be expressed with microprogram related labels such as "Overflow Status Available" rather than "OSCLK". Much of this information may not be used until code is actually written although it may influence design.

The second design aid is a microcode simulator. A simulator is typically used during the checkout and debug stages of microcode development. If it produces an accurate simulation of microcode execution, it can also be used during design to provide insight into device anomalies that might seriously affect hardware design as well as microcode design.



Note: This timing chart is for hypothetical overlapped microinstructions in which the execute phase of instruction n overlaps the fetch phase of instruction n + 1.

Figure 8 - Worst Case Timing for Microinstruction Events

2.3.2.2 CODING TOOLS AND TECHNIQUES

Although much research is being done in the area of Higher Level Languages (HLLs) or Higher Order Languages (HOLs), optimization techniques, transportable languages and other vehicles for aiding the microprogrammer's task , an assembler is the most utilized means of directly translating source code into executable binary information.

The typical microassembler has many of the same features as software assemblers:

a. Labels to define locations.

b. Provisions for symbolically defining variables.

c. Provisions for inserting or appending comments.

d. Numberbase (Radix) specification.

e. And others.

If the microinstruction has a diagonal or horizontal format, individual microoperations are normally defined mnemonically. A good assembler can check for conflicting microoperations and label them as errors.

There are self documenting assemblers in use today. These assemblers generate machine dependent comments that describe a particular microinstruction. Two such assemblers are described by Laws [5] and Firth [2]. The comments describe the effects microcode has upon the hardware, and are not intended to replace the programmer's comments relating to program flow. Firth states that the commenting assembler produces a very large listing and runs slowly, however, it is used mainly to compile microcode to be released to manufacturing and field service personnel. Early reports indicate that even though the MV/8000 microcode is much more complex than previous Data General microcode, training times are not proportionally longer.

One software development technique which could have application in the microprogramming field is a program representation called "GREENPRINT" [7]. This approach combines a diagrammatic representation of a program with its (the program's) source listing on the same page.

This technique could be manually implemented, and could add significantly to the understanding of microcode.

Some hints passed on by microcode developers:

a. Assembly listings should be clean and uncluttered.

b. A simple microinstruction should be contained on one line for clarity.

c. Have uniformly formatted fields, free form source is not as easily understood.

One additional technique which is helpful in the coding of complicated horizontal instructions, is the utilization of the Data Dependency Graph. This graph portrays each microoperation as a unique node. The arcs connecting the nodes indicate which microoperations are directly data dependent on other microoperations. Any node is data dependent on its immediate predecessors and is data independent of nodes for which it is neither a predecessor or a successor.

An interesting approach toward a machine independent microprogramming language is described in a paper by Patterson, Lew and Tuck [3]. The paper describes a low level language called YALLL (Yet Another Low Level Language).

This approach was to develop a low level language as an intermediate step in HLL to microcode production.

> ". . .: Rather than to try to bridge the gap from HLL to microarchitecture in one step, we have designed a low level language that is capable of producing microcode for different machines. Hopefully, it should be easy for compilers to generate low level language code and easy for optimizers to translate this low level language code into efficient microcode."

YALLL was used to produce microcode for 2 different micro-architectures (The Digital Equipment Corp VAX 11/780 and the Hewlett Packard HP 300). Although YALLL was not able to produce code equivalent to hand optimized code, the results were good. The main point to be made, however, is that programs developed using the YALLL approach are easier to read and write than those developed using typical microassemblers.

2.3.2.3 CHECKOUT AND DEBUGGING AIDS

Due to the inherent difficulty in checking out and debugging microcode on devices that in many cases have little or no operator interface capabilities, the simulator has proven to be one of the most important tools in the microcode development process. A simulator duplicates the function of microinstruction as they would be executed on the real machine. It usually has facilities for user interaction with the microprogram that is being simulated. An additional benefit is provided when new hardware is being developed. Both microcode and hardware development can proceed in parallel, thus saving valuable time. When the simulator is written in a register-transfer language (or another hardware oriented language) it can also be used to uncover hardware design errors.

In a paper on the simulation of microprogrammed LSI circuits, Nielson [8] states:

> "Through simulation, all elements (even those internal to an LSI chip) can be made visible on a cycle-by-cycle basis. This capability is advantageous in detecting potential hidden design defects. For example, routine A may inadvertently alter a register during its execution. The effects of this error may not affect operations until sometime later in the execution of the microprogram. Such problems are easily traced in a simulation, but are very difficult to trace by other means."

Firth [2], in his paper on the tools used in development of the MV/8000 microcode states:

> "Simulation played a significant role in the development of the MV/8000 microcode. The most significant factor being the early availability of microcode and diagnostics with a high confidence factor. But also of great importance is the estimate that the simulation effort saved 3 months in the development cycle of the MV/8000 . . . while some bugs in the microcode and diagnostics did appear, actual experience showed that they accounted for less than 10% of all errors encountered in the lab".

A simulator should have some, if not all, of the following features:

a. Breakpoints - The user can set one or more breakpoints in the executable microcode so registers, stacks and other simulated hardware elements may be examined.

b. Data Modification - The user can (after encountering a breakpoint or data trap) examine and/or modify any of the hardware elements which contain data.

c. Trace - The user may specify that particular registers, buses or hardware elements be monitored and displayed for each microinstruction.

d. Time Keeping - Execution time of the simulated microcode is kept and displayed during Traces.

e. Data Traps - Data traps provide a capability similar to breakpoints. The simulator can monitor user designated registers or buses and cease simulation upon detection of specified data values.

2.3.2.4 MICROPROGRAM VERIFICATION AND TEST

Verification is the process of proving that a microprogram is error free while test is the process of uncovering errors in that microcode. These errors can be mistaken in the code itself, a conceptual error in the design or an error with respect to fulfilling the original requirements.

Verification, using proofs of correctness, has been described in a paper by Joyner, Carter and Leeman (9). Although much research has and is being done in the area of verification, it is not used by microcode developers.

Test, according to the survey, is in no way standardized. It tends to be project oriented, due to the various hardware configurations that may be used or developed from project to project. As described in section 2.3.2.3, there is an inherent difficulty in checking out and debugging microcode. These same difficulties make thorough testing awkward at best. Testing may be performed using a simulator such as described in the previous section. This technique is suspect, however, since it assumes that the simulation is a perfect respresentation of the hardware. If possible the microcoded device (including the microcode) should be designed with testing in mind.

Testing should be well planned, documented and executed. Software techniques may be utilized with respect to values, ranges, bit patterns and other data sensitive testing. Timing and synchronization or coordination tests between hardware and/or microcoded devices must utilize measurement equipment or logic analyzers. It is wise to use personnel who have not been involved in the microcode development to perform formal testing.

3.0 MICROCODE DEVELOPMENT

Early implementations of software were literally bottoms up in that requirements were ill defined, often verbal and poorly communicated to the programmer. Integration and testing very often brought software packages together for the first time and in some cases software and hardware were also brought together for the first time. It became apparent that doing detailed design and/or coding without a carefully thought top level design usually resulted in a poor product. Integration of software and hardware under these circumstances invariably produced unwelcome surprises with necessary patches or recoding of some areas.

The modern top-down approach starts from a top level requirements definition and proceeds "downward" to functionally distinct program and data modules. Through systematic refinements it produces more and more detail within the design which can be reviewed and analyzed at each step with respect to the functional requirements, completeness, etc. All of this design review is prior to the coding effort. The entire process, if performed correctly, can actually force requirements to be well formed in the sense of clarity and completeness. The advantages of the top-down approach are equally valid for microcode development.

3.1 THE MICROCODE DEVELOPMENT PROCESS

In previous sections of this report an effort was made to illustrate the wide variance between microcode architectures and their complexities. When these variances are coupled with the range of microprogramming effort (is it 50 instructions or 2000?), it becomes obvious that a rigidly enforced development process as described here may not produce the most cost effective result. For example, a system which utilizes only one microcoded device such as a sequencer with only 100 vertical micro-instructions should not require the same level of documentation as a simple control device with 500 to 1000 horizontal instructions. Each phase in the development process should be performed with documentation commensurate with the microcode effort. Documentation necessary for operations and maintenance, however, should be finely and thoroughly detailed for simple as well as complex microcoded devices.

The development process documentation described in the following sections assumes that a large scale microprogramming effort is being undertaken. Proposed document section headings are preceded by an asterisk (*) so as not to confuse them with the section headings of this report.

One of the tasks of this project was to review existing military documentation standards for applicability to the microcode development process. There is an intentional similarity between the proposed documentation and some portions of military standards where the microcode and software development processes are in agreement.

It is felt that where both microcode and software are being developed for a single device or system, a single set of controlling documents could suffice for both. The development process consists of a number of well defined phases or steps. These steps are:

a. Analysis - Planning and the development of functional requirements.

b. Design - Transform requirements into a design approach.

c. Code and Checkout - Coding and debugging to implement the design approach.

d. Test and Integration - Components integrated and formally tested against requirements.

e. Operations and Maintenance - Hidden errors are corrected and new capabilities or requirements may be implemented.

A more thorough description of these steps follows:

3.1.1 THE ANALYSIS PHASE

The analysis phase starts with the formulation of a Microcode Development Plan (MDP) or Computer Microcode Development Plan (CMDP) in the case of a microcoded computer. A recommended format for such a plan follows. At a minimum the plan should address each item with a statement as to why such an item is not applicable to the development effort for which the plan is being formulated.

3.1.1.1 THE MICROCODE DEVELOPMENT PLAN

*1.0 INTRODUCTION

Indicate the scope of the development effort, the purpose, application, and authority for the project. Give an overview of management philosophy and methodology.

*2.0 APPLICABLE DOCUMENTS

List the documents that this plan is responsive to or that will be utilized in implementing the provisions of this plan.

*3.0 ORGANIZATION AND RESPONSIBILITIES

The MDP shall provide a delineation of the contractor's organization structures, authorities, responsibilities, interfaces, skill requirements and lines of communication necessary to manage and execute the scheduled activities to assure proper task completion.

The MDP shall include the relationship assumed between the contractor and independent or redundant verification and validation groups, if they exist, other contractors, subcontractors, the procuring agency, the support agency and the user agency. The MDP shall also include identification of skill requirements and key managers and employees by name.

*4.0 DEVELOPMENT PHASE PLAN

The MDP shall describe the development phases required to achieve the objectives of the microcode development effort. Identification of the necessary development steps such as planning, requirements, design, code and checkout, qualification test, systems test and integration, and operations and maintenance for each Microcode Program Configuration Item (MPCI) and their relationship to the contractor's Work Breakdown Structure (WBS) shall be included. Tasks associated with support functions (such as documentation, configuration management, management reviews, quality assurance etc.) and their relationship to other contractual tasks shall be described.

*5.0 TEST PLAN

The MDP shall provide a presentation of the integration and test philosophy and approach for Microcode Program Components (MPC) and MPCIs, how the philosophy is applied in the design and scheduling, and how the approach leads to the Preliminary and Formal Qualification Tests. The MDP shall provide a definition of the interfaces and responsibilities (such as training, if required) among test groups and other project participants. It shall also identify resource and validation activities. The MDP shall also provide procedures for the coordination and support of MPCIs and other computer resources during formal system testing, including necessary training and transfer of data.

*6.0 CONFIGURATION MANAGEMENT PLAN

The MDP shall provide the special aspects of computer microcode configuration management not addressed in the overall Configuration Management Plan. It shall include the organizational placement, change, deviation, waiver authorization and control, internal engineering change board, configuration identification, microcode program library control, the control or changes to any non-deliverable microcode programs, the handling of interface control between the hardware and the microcode of the system, maintenance of independent master tapes or disks, control of development versions of the MPC and MPCIs, handling of source and binary microcode, control over test environments, and relationships to Quality Assurance functions.

*7.0 QUALITY ASSURANCE PLAN

The MDP shall describe the procedures and organization that will be used to satisfy all the microcode QA requirements. This shall include managing responsibilities, reviews and audits, controls, discrepancy reporting and corrective action, and test assurance to assure completeness, validity, traceability to requirements, testability and compliance.

*8.0 DOCUMENTATION

The MDP shall provide a description of the contractor documentation practices (as they apply to microcode) and a description of the procedures which will be used to keep the informal documentation current. For each project the customer's Contract Data Requirement List (CDRL) should be carefully examined for documentation that will supersede normally stated documentation requirements.

*9.0 TRAINING AND DEPLOYMENT

The MDP shall identify the recommended training, support philosophy, and the resource requirements for use after the full-scale engineering development phase. This section of the MDP shall summarize or specifically reference the plan for the transfer of computer resources, including support software or microcode tools to the appropriate customer or agency.

*10.0 REVIEWS AND REPORTING

The MDP shall identify the reviews and reporting that occur during the microcode life cycle.

*11.0 DEVELOPMENT RESOURCES

The MDP shall identify the allocation of facilities, laboratories, computer time, test equipment, and other relevant resources against the organizational structures, work elements, and schedules. It shall identify project-peculiar resources required such as special purpose hardware and computer programs, government furnished items and special data. Also a list of items which may impact resources such as high risk development items, special security requirements, and subcontractor control shall be identified.

*12.0 ENGINEERING STANDARDS

The MDP shall provide a definition or identification of software engineering practices as they apply to each MPCI and computer resource support product, and how these practices will be assured. Examples of engineering practices include standards, input/output standards (when applicable), guidelines for microcode program subdivisions, coding techniques, programming language and other disciplines affecting development.

*13.0 REQUIREMENTS AND RISK ASSESSMENT

The MDP shall identify potential problem and high risk areas in terms of cost, schedule and technological risk.

*14.0 SCHEDULES

The MDP shall show the development schedule, major milestones required by contractual agreements and significant internal milestones. Also, critical paths and potential critical paths which may occur due to unplanned schedule slippage shall be identified. A recommended life cycle overview with appropriate milestones shall be furnished.

*15.0 GOVERNMENT FURNISHED EQUIPMENT

The MDP shall list all equipment, services, facilities and programs required for the production or test of the software or microcode, which are to be furnished by the Government. All necessary conditions should be stated, such as schedule of usage and training/maintenance support required.

(*Refers to numeric organization within the MDP document).

3.1.1.2 THE REQUIREMENTS DOCUMENT.

Once the MDP has been accepted, the next step in the requirements definition phase begins, that of formal requirements definition. The requirements documentation contains the operational, functional and interface requirements when applicable, and the quality assurance provisions that are necessary for development and testing of the operational microcode. A recommended format for such a requirements document follows. At a minimum, the document should address each item with a statement as to why such an item is not applicable to the development effort for which requirements are being established.

The recommended requirements document format and content is:

*SECTION 1.0 SCOPE

*1.1 Identification. This paragraph shall contain the approved identification, nomenclature, and abbreviation for the microcode. This section of the CI specification shall begin with the following opening phrase: "This part of this specification establishes the requirements for performance, design, test and qualification of a microcoded program identified as (insert nomenclature and configuration item number).

*1.2 Functional Summary. This paragraph shall contain a summary of the purpose of the specification and a brief description of the overall microcoded program by major functions or tasks. It shall further identify and summarize the specification content, composition and intent.

*SECTION 2.0 APPLICABLE DOCUMENTS

This section shall list all customer, subcontractor or other documents which are applicable to this requirements specification.

*SECTION 3.0 REQUIREMENTS

This section shall contain performance and design requirements for the MPCI and establish those requirements which normally will be verified during category I or equivalent test (see section 4 of this document). This section shall also define the MPCI and specify design constraints and standards necessary to assure compatibility of the MPCI with other microcode programs and equipment. Performance and design requirements to be included herein shall be allocated from, identical with, or in recognition of, requirements established by the system/system segment specification. Requirements included in the system/system segment specification, which are directly related to requirements specified herein, may be incorporated by reference. Requirements shall be specified to the level of detail necessary to establish limits for design. Quantitative requirements shall be within the three principal subparagraphs included herein. The introductory paragraph shall include a general description of the MPCI and its functions within the system/equipment to which it applies.

*3.1 Microcode Definition. This paragraph shall in subparagraphs included herein, specify the functional relationship of the MPCI to other microcode programs, software or hardware.

*3.1.1 Interface Requirements. This paragraph shall specify, either directly or by reference, requirements imposed on the design of the MPCI because of its relationship to other microcode, software or hardware equipment.

Interfaces defined in this section shall include, at a minimum, all relevant characteristics of the microcoded device, such as memory word size, control word size, control store characteristics and structure, timing considerations, interrupt capabilities and any special hardware features. The device characteristics may be described by reference to the applicable documentation and descriptions. If a known compiler is to be used, the programming language shall be specified as one of the interfaces in subparagraph 3.1.1.2. If the compiler or assembler is to be constructed as part of the development of this MPCI, the language characteristics shall be defined under paragraph 3.2. Detailed Functional Requirements.

*3.1.1.1 Interface Block Diagram. The relationship of the MPCI to other microcode computer programs, or hardware with which it must interface shall be graphically portrayed in this paragraph. This paragraph shall incorporate, in subparagraphs as appropriate, a functional block diagram or equivalent representation of the interface requirements of the MPCI. The graphic portrayal of the MPCI shall be accomplished to the level of detail necessary to identify the functional interfaces and timing coordination between the MPCI and other identified MPCIs, software programs, and hardware.

*3.1.1.2 Detailed Interface Definition. This paragraph shall specify, in subparagraphs as appropriate, the functional relationship of the MPCI to interfacing software or other microcode. This information shall be given in quantitative terms with tolerances where applicable to the level of detail necessary to permit design of the MPCI. Functional interfaces shall specify the input/output requirements of the MPCI in terms of data rate, data format, etc. In addition, this paragraph shall specify design requirements imposed upon other microcode or computer software as a result of the design of this MPCI.

*3.2 Detailed Functional Requirements. This paragraph shall specify in subparagraphs defined below, the functional requirements of the MPCI. Requirements shall be stated in quantitative terms, with tolerances where applicable. General and descriptive material may be included in basic paragraph 3.2, which shall incorporate either directly or by reference, a functional block diagram or equivalent representation of the MPCI. The graphic portrayal shall be accomplished to the level of detail necessary to illustrate the functional operation of the MPCI, the relationships between these functions and the relationships between the functions and other identified system/equipment functions. This diagram is not intended to be restrictive on microcode detail design. Requirements for separately identified MPCI functions shall be described in subsequent paragraphs as appropriate. A subparagraph shall be included for each operational function, plus special functions such as sequencing control, error detection, I/O control, diagnostics, etc. The descriptions of these MPCI functional requirements shall include the relative sequencing, periodicities, options, and other important relationships of each as appropriate. Paragraphs 3.2.X and subparagraphs shall be repeated for each function above.

*3.2.X <u>Function X</u>. The basic paragraph for each function shall begin with descriptive and introductory material which defines the function and its relationship to other functions. Then, the following three subparagraphs shall specify the quantitative requirements concerning the function.

*3.2.X.1 <u>Inputs</u>. This paragraph shall specify either directly or by reference to another part of this specification the source(s) and type(s) of input information associated with a function of the MPCI. This shall include a description of the information, its source(s) and, in quantitative terms, accuracy/precision requirements and the performance required in processing inputs.

*3.2.X.2 <u>Processing</u>. This paragraph shall provide a textual and mathematical or logical description of each of the processing requirements of each function. Presentation of the descriptions under each function shall include:

a. Purpose - This area shall describe the exact intent of the operation(s). This involves a definition of the specific input and output parameters and the processing required.

b. Approach - This area shall contain a textual description of each mathematical or logical operation specified. The accompanying narrative shall identify accuracies required, sequence and timing of events, and relevant restrictions or limitations. Derived equations shall be shown with appropriate mathematical, logical and control symbols adequately defined.

*3.2.X.3 <u>Outputs</u>. This paragraph shall specify, either directly or by reference to another part of this specification, the destination(s) and type(s) of output information associated with a function of the MPCI as a result of the processing described in paragraph 3.2.X.2. This shall include a description of the information; its destination(s); and in quantitative terms, accuracy/precision requirements and performance required within the function for outputs where applicable.

3.2.n <u>Special Requirements</u>. This paragraph shall specify, in appropriate subparagraphs, requirements which affect the design of the MPCI and are distinguishable from performance requirements of the prior subparagraphs of paragraph 3.2. These requirements result from general considerations of MPCI usability. These may include, but are not limited to, requirements for:

a. The use of microcode programming standards to assure compatibility among microcode components.

b. Microcode organization, such as overall program segmentation. In addition, for MPCIs which contain or process classified information, special attention shall be given to the requirements for protecting classified information.

c. Microcode design resulting from considerations of modifications to the MPCI during operation (e.g., on-sight modification requirements and the permissible amount of operational degradation allowed during installation of modification (if applicable) may be specified).

d. Special features, to facilitate the testing of the MPCI. For example, special procedures for the design of Microcode Program Component (MPC) interfaces to provide traps for trace lists etc, where appropriate.

e. Expandability (growth potential) to facilitate modifications and additions to the MPCI.

n* - The next sequential number following the number of the last function (n = last X + 1).

*3.2.n.1 Human Performance. Human performance/human engineering requirements for the MPCI shall be specified in this paragraph. As may be the case with microcode, human performance/human engineering requirements may not be applicable and should be so stated as necessary.

*3.3 Data Base Requirements. These paragraphs shall specify, in descriptive and quantitative terms, any data base requirements which affect the design of the MPCI.

*SECTION 4.0 QUALITY ASSURANCE PROVISIONS

Requirements for formal verification of the performance of the MPCI in accordance with the requirements of section 3 of this specification shall be specified in this paragraph. Formal verification of performance of the MPCI shall determine acceptance of the MPCI. This paragraph shall specify formal verification requirements to a level of detail which:

a. Designates verification requirements and methods in section 4 for performance and design requirements in section 3. The methods of verification to be specified herein may include inspection of the MPCI, review of analytical data, demonstration tests, and review of test data.

b. Specifies requirements for verification to the level of detail necessary to clearly establish the scope and accuracy of the test method.

c. Permits ready identification of each verification requirement specified in section 4 with the appropriate performance/design and requirement paragraph in section 3.

d. Allocates verification requirements to the subparagraphs included herein.

NOTE: This section shall not incorporate, either directly or by reference, detail test planning documentation and operating instructions. Requirements specified herein shall be the basis for preparation and validation of such documents.

*4.1 Introduction. This paragraph shall establish the requirements which provide the basis for development of a test plan and test procedures for the subject program. All test/verification requirements shall be specified within the subparagraphs included herein.

*4.1.1 Category I Test. The term "Category I Test" as used herein is defined to include all testing of the MPCI other than that accomplished during the formal category II (or equivalent) system/configuration item test programs. (See paragraph 4.1.5 below.) Category I testing is subdivided into the following broad types:

a. Microcode test and evaluation - Tests conducted prior to and in parallel with Preliminary or Formal Qualification Tests. These tests are oriented primarily to support the design and development process.

b. Preliminary Qualification Tests - Formal tests oriented primarily towards verifying portions of the MPCI prior to integrated testing/Formal Qualification Tests of the complete MPCI (see paragraph 4.1.3 below). These tests will typically be conducted at the contractor's design and development facilities.

c. Formal Qualification Tests - Formal tests oriented primarily towards testing of the integrated MPCI, normally using operationally configured equipment at the Category II site prior to the beginning of Category II testing. This testing will emphasize those aspects of the MPCI performance which were not verified by preliminary tests. The testing requirements which cannot be verified during Category I Test shall be specified in paragraph 4.1.5.

NOTE: Requirements for verification included in the system/system segment specification, which are directly related to requirements specified herein, may be incorporated herein by reference to avoid redundant establishment of the requirements.

*4.1.2 Microcode Test And Evaluation. This paragraph shall contain the following:

Microcode test and evaluation which satisfies one or both of the criteria listed below shall be included herein. (Routine tests accomplished in support of design and development, which do not satisfy one or both of these criteria, shall not be specified herein.)

a. They are intended to be the only source of data to qualify specific requirements in section 3.

b. They must be accomplished as part of an integrated test program involving other systems/equipment/computer programs (e.g. verification of requirements in paragraph 3.1.1).

*4.1.3 Preliminary Qualification Tests. This program shall specify only those preliminary qualification test requirements which are oriented toward verifying proper performance of portions of the MPCI prior to integrated testing of the complete MPCI. Testing accomplished in support of design and development shall not be specified herein. Requirements for preliminary qualifications specified herein shall reference requirements in section 3.

*4.1.4 Formal Qualification Tests. This paragraph shall specify requirements for Formal Qualification Tests of the integrated MPCI to demonstrate and/or verify that the requirements established in section 3 have been satisfied. This paragraph shall, in the subparagraphs as appropriate, specify the requirements and method of verification for the requirements specified in section 3, with the following exceptions:

a. The requirement in section 3 has been identified, and verification that it has been satisfied by one of the tests included in paragraphs 4.1.2 and 4.1.3.

b. The requirement in section 3 is peculiar to Category II type system testing and will be identified in paragraph 4.1.5.

Verification of the requirements may be accomplished by inspection, demonstration, test, and review of test data, or combinations of these. This paragraph shall contain a subparagraph for each of the principal methods of verification, and shall specify therein the requirements of section 3 to be verified by the method.

*4.1.5 Ca‘agory II System Test Programs. This paragraph shall identify requirements specified in section 3 which cannot be verified until Category II esting (or equivalent) and must be listed as a Category II Test requirement.

*4.2 Test Requirements. This paragraph shall specify the requirements for each type of testing. The requirements shall include test formulas, algorithms, techniques and acceptable tolerance limits, as applicable.

*SECTION 5.0 NOTES

This section shall include information which is stated here for administrative convenience only, and is not a part of the specification for the MPCI in the contractual sense (i.e., it shall not include requirements which constrain design, development, and qualification of the MPCI and require compliance by the contractor. The text may be preceded with the statement "Administrative Information Only - Not Contractually Binding." This section of the specification shall include information of particular importance to the procuring activity in using this particular specification as a contractual instrument for acquisition of the MPCI either initially or for follow-on procurement.

Background information or rationale which will be of assistance in understanding the specification itself or using the MPCI it specified, may be included herein (e.g., technical data ordering instructions).

*SECTION 10.0 APPENDIX I.

This section of the specification shall contain requirements which are contractually a part of the specification but which, for convenience in specification maintenance, are incorporated herein (e.g., requirements of a temporary nature or for limited effectivity). Appendixes may be bound as separate documents for convenience in handling (e.g., when only a few parameters of the program are classified, an appendix cont 'ning only the classified material may be established). Where parameters are placed in an appendix, the paragraph of section 10 shall be referenced in the main body of the program specification in the place where the parameter would normally have been specified. Typical data that may be included in [illegible]crocode development specification appendixes include:

a. Mathematical, logical, or control derivations;
b. Alternate method;
c. Summary of equations;
d. Definition of terms.

(*Refers to organization of Requirements Specification only).

3.1.2 MICROCODE DESIGN SPECIFICATIONS

3.1.2.1 PRELIMINARY MICROCODE DESIGN

After the system allocated baseline design has been established in the MDP, a preliminary microcode design is developed and may be documented in a draft version of a MPCI design specification. The draft version of the specification will contain the assignment of each of the microcode requirements to a specific functional microcode module and the functional interface for each module. Design estimates of program storage and processing time requirements are presented along with program functional diagrams.

3.1.2.2 DETAILED MICROCODE DESIGN

After approval of the preliminary microcode design, a detailed microcode design is developed. The detailed microcode design may be initially documented by detailed program flow charts or the equivalent. These detailed design charts are then integrated into the draft version of the MPCI design specification, to provide a "code-to" version of the specification. The baseline version of the specification should contain all of the information necessary to proceed with program coding and checkout.

During the microcode code phase, the detailed microcode design is implemented. This design may be documented by updating the MPCI design specification to include the design implementation which has been built into the operational microcode. The program listing can either be added as an appendix to the updated MPCI design specification, or included in a separate program package. The documentation produced during this phase presents a complete description of the "as-built" microcode. The recommended format and content for an MPCI design specification is provided:

*SECTION 1.0 SCOPE

This section shall contain the following lead phrase, "This specification establishes the requirements for complete identification (insert configuration item number and nomenclature)."

*SECTION 2.0 APPLICABLE DOCUMENTS

List those documents (specifications, standards, bulletins, manuals, etc.) which are applicable to paragraphs within other sections of the specification. Within the body of the specification, reference to these documents shall be to their basic document number and to the title, specifically identified requirements, or other definitive designation.

*SECTION 3.0 REQUIREMENTS (TECHNICAL DESCRIPTION)

This section shall specify the detailed configuration of the MPCI. It shall contain a complete technical description of the MPCI structure and functions, the data base, and the individual microcode program components (MPCs). The rules established for assigning symbolic names and register usage conventions to components within the CI shall be delineated. General and descriptive material may be included in the basic section 3 lead paragraph.

*3.1 Functional Allocation Description. This paragraph shall contain a description of the overall structure and functions of the MPCI. This description shall include the allocation of functions (as delineated in the requirements specifications) to the MPCs that comprise the MPCI. If the MPCs are grouped into functional entities (packages) for separate stages of development and checkout, this grouping shall be delineated.

*3.2 Functional Description. The individual microcode program components (MPCs) shall be described in separate paragraphs as required. This description shall be given at a level of detail that will define the design and configuration of the MPC sufficiently to allow for MPC modification and adaption in the operation phase. Each MPC shall be described in words, flow charts (or equivalent), and a listing of the instructions used. The basic paragraph 3.2 shall contain the following lead phrase, "This paragraph contains the detailed technical descriptions of the microprogram components identified in paragraph 3.1 of this specification. The instruction listing contained in paragraph 3.2.1.6 specify the exact configuration of the (name of MPCI)." The following subparagraphs shall be repeated for each MPC:

*3.2.N Microcode Component N. The basic paragraph shall identify the MPC by including, as a minimum, the title, tag (symbolic code), and MPC identification number. It shall also include a brief abstract of the tasks of the MPC, the language in which it is written, and its major functional interfaces. The component shall be described in detail in subparagraphs.

*3.2.N.1 MPC No. N Description. This subparagraph shall describe in words, figures, equations, and references to the flow charts (or equivalent) of subparagraphs 3.2.N.2, the operations and design of the MPC. This paragraph shall contain, as appropriate, a description of the program logic and data flow; equations to be solved; algorithms used to solve these equations; timing and accuracy characteristics; and any special conditions for operation of the MPC. The description shall be sufficiently detailed to facilitate understanding of, and modification to, the listing given in subparagraph 3.2.N.6. Equation derivations and numerical analysis shall not be included herein, but may be included in section 6 (Notes).

*3.2.N.2 MPC No. N Flow Chart. This subparagraph shall graphically portray the operations performed by the MPC. This shall be done by a (series of) flow chart(s) (or equivalent) which depict the processing described in subparagraph 3.2.N.1, including the sequence of operations and decision points, in the MPC. The highest-level chart shall depict on a single sheet the overall information flow of the MPC and shall reference the charts in paragraph 3.4 that identify the MPC. In general, the lowest-level chart identifies all decision points in the MPC and references higher level charts as appropriate. All charts shall use descriptive symbology and shall reference the program listing of the MPC by use of statement labels or tags.

*3.2.N.3 MPC No. N Interfaces. This subparagraph shall describe the relationship of the MPC to other MPCs, to that part of the data base external to the MPC and where applicable to other MPCIs. At a minimum, this subparagraph shall include, either directly or by reference to other paragraphs of the part II specification, the following:

a. The set-use matrix, table, item, buffer definitive information and input/output format information.

b. Unique to MPC interfaces that, because of their nature, composition, use or meaning, require an individual definition.

*3.2.N.4 MPC No. N Data Organization. This subparagraph shall contain or refer to a portion of subparagraph 3.2.N.6, if appropriate; contain a list and description of all data items and tables which are unique to and included within the MPC; and describe the areas of memory available for temporary storage. This list shall include all internally defined symbols and constants and their equivalents and meaning.

*3.2.N.5 MPC No. N Limitations. This subparagraph shall summarize any known or anticipated limitations of the MPC. A listing of all restrictions and constraints which apply to the MPC shall be provided, including timing requirements, limitations of algorithms and formulas used, limits of input and output data, associated error correction sensing, and the error checks programmed into the routines.

*3.3 Storage Allocation. The relationship of the MPCI storage requirements to the total computer equipment storage capability shall be graphically portrayed in subparagraphs as appropriate, either directly or by reference, a schematic diagram or equivalent representation. This graphic portrayal of the MPCI storage shall be accomplished to the level of detail necessary to identify such requirements as data base, microprogram, microprogram operation, and spare storage allocations. In addition, the timing, sequencing requirements, and equipment constraints used in determining the allocation shall be described. If allocations cannot be specified precisely or portrayed graphically in a manner meaningful for program design, the algorithms used to allocate storage shall be included.

*3.4 Microcode Program Functional Flow Diagram. This paragraph and subsequent subparagraphs shall show the general system flow of both data and control. The main paragraph 3.4 shall graphically portray the operations performed by the MPCI by a (series of) chart (s) which depicts the processing being performed, the sequence of operations, and decision points. A top-level chart shall be used to depict in a single figure the overall information of the MPCI. This diagram shall reference lower level charts included in this paragraph, as appropriate, to provide more detailed information. The lowest-level charts shall be those which identify as functional entities the microcode program components described in section 3.2 above. If the MPCI is designated to operate in more than one mode, each mode shall be clearly distinguishable in both the text and diagram(s).

*3.4.1 Program Interrupts. This paragraph shall list all program interrupts and describe their effect on the design of the control logic. Each interrupt shall be fully described as to source, purpose, type, and the required response. The probable rate of occurrence of interrupts shall also be given, if available. This description shall be in the functional terminology developed in paragraph 3.1 of the specification.

*3.4.2 Logic of Subprogram Reference (Component Reference). This paragraph shall contain a detailed description of the control logic involved in referencing each MPC. Timing and sequencing of operation of the MPCs relative to each other shall be discussed. If the sequencing is dynamically controlled during the MPCIs operations, this description shall include the method for sequence control and the logic and input conditions of that method. Such factors as timing variations, priority assignments, plus such internal operations as data transfer in and out of memory, sensing of discrete input signals, and the timing relationships between interrupt operations within the MPCI shall be included.

*3.4.3 Special Control Features. This paragraph shall describe all the special control features that affect the design of the control logic but are not part of the normal operational functions (e.g., system loop tests for routine maintenance).

*SECTION 4.0 QUALITY ASSURANCE

This section shall reference the test plans and procedures that were used for the qualification of the MPCI during category I testing. In addition, this section shall also reference and/or specify those tests which apply to duplication of MPCIs.

*4.1 Test Plan/Procedure Cross Reference Index. In this subparagraph, the relationship of the functions to their test plan and test procedure documentation shall be referenced by cross indexing. This subparagraph shall also reference any special test tools or capabilities (e.g., tapes, ROMS, PROMS etc.) which are covered in this specification.

*SECTION 5.0 PREPARATION FOR DELIVERY

This section shall specify, in subparagraphs as appropriate, the requirements for packaging, marking, and otherwise preparing the MPCI for shipment and storage (if applicable).

*5.1 Preservation and Packaging. These paragraphs shall describe the preparation for delivery requirements of the completed MPCI. Packaging requirements include a description of the product packaging (ROMS, PROMS, EPROMS, disks, manuals), preservation methods, packing, etc., involved in preparing the MPCI for shipment and storage. Special handling requirements or other special considerations shall be included (e.g., when shipping magnetic storage media, special EMI requirements shall be delineated.

*5.2 Markings. This subparagraph shall specify in detail the identification markings to appear on packages that make up the MPCI to be delivered.

*SECTION 6.0 NOTES

This section of the specification is not contractually binding, and therefore should not be used for contractual requirements. It shall include administrative or background information (e.g., ordering instructions for technical data pertaining to the microcode program, or specific information related to the use of the program in future assembly and integration testing). It shall not include requirements which constrain design, development, or qualification of the MPCI. The text may be preceded with the statement "Administrative Information Only - Not Contractually Binding." It shall reflect the technical manuals which can be singularly and peculiarly identified with the MPCI, and which are necessary to its operation and maintenance. For each MPC, this paragraph shall include any pertinent information not included in the above subparagraphs, such as rejected alternative MPC designs, the rationale behind the design, reference material in support of the algorithms used, and suggestions for future modifications to the MPC if changes in requirements should materialize. It shall also describe, as appropriate, the pertinent tests which were performed to verify the final implementation of the MPC, with key test results included or referenced.

*SECTION 10.0 APPENDIX I

This section of the specification may contain the MPCI listing and requirements which are part of sections 3 and 4, but are bound separately for convenience. Examples are computer-produced listing set-use tables (variables are defined and instructions which set or use that variable are listed), multisite adaptation requirements, etc.

(*Refers to organization of the Firmware Program Configuration Item only).

3.1.3 CODING STANDARDS

The following paragraphs assume that as a minimum a microcode assembler has been used to generate the binary microcode. If an assembler is not available and the device has been judged to be very simple in the sense of length and complexity of microcode and may therefore be coded directly in binary, octal, or hexadecimal characters, the programmer should be required to write "Assembly like" commands. The command language should be documented so that an assembler could be developed and utilized at a later date.

In order to facilitate the understanding and maintainability of microcode a number of paragraphs shall be included with each module.

a. Each module shall have a title. The title should be meaningful with respect to the modules function. A title such as "PARITY CHECK" is preferable to "CHECK P" or "FUNCTION 1" or "F1".

b. Each module shall have comments or statements explaining:

1. The module's functional description (a few sentences should be sufficient).

2. The module's purpose, use, and processing activities. Technical details should not be included.

3. Inputs. Any variables, tables, data and status registers, stacks or other data input sources shall be identified as to type, size (bites, bits, fields, etc), value limits, accuracy or precision requirements, or timing with respect to frequency of arrival or availability.

4. Outputs. Variables, tables, data and status registers, etc., shall be identified and described in the same manner as inputs.

5. Registers used. Registers which have been accessed or modified shall be identified and their end conditions (contents) noted. Status register and/or bits shall be included.

6. Subroutines or Submodules called. If the microcoded device has "Subroutine Call" capabilities, any calls to subroutines or submodules shall be listed.

7. Limitations. Any contraints upon the module's execution shall be described. Critical timing considerations whether they are imposed by a system constraint or by other modules shall be noted. Present minimum, maximum and nominal execution times shall be noted.

8. Modifications. A chronological list of modifications shall be maintained. Each modification or group of modifications performed for a revised module shall be described together with a date and the responsible individual.

9. Special Comments. If the microcode is subtle or confusing it may require some special explanation in order to make the code easier to understand.

c. If there is a pseudo code or other program design language material for the module, it shall be listed.

d. If there is a logic or flow diagram in graphic form it shall be included.

e. The microcode listing itself shall be included. If there are line-by-line comments they shall be presented in an orderly fashion, columnized and set to the side so as not to impose on the instruction. If there are not line-by-line comments, the module shall be broken up into functional groups and comments attached to each group explaining the groups process. These comments shall be in addition to any "self documenting" assembler notations which are usually restricted to hardware related functions.

3.1.4 MICROCODE TEST DOCUMENTATION

3.1.4.1 TEST PLANS AND SPECIFICATIONS

The MPCI Test Plan is developed from the Quality Assurance section (4.0) of the MPCI specification and from information contained in the Microcode Development Plan. The MPCI Test Plan defines the scope of tests required to ensure that the program meets all applicable technical, operational, and performance specifications. It establishes detailed acceptance criteria for the program and identifies each level of testing. It also indicates the purpose, relationship to other tests, and system environment for each level of test. The recommended format and content for an MPCI Test Plan is provided.

*SECTION 1.0 PURPOSE

This section shall state the purpose of the MPCI Test Plan and shall identify the MPCI to which it applies by number and approved nomenclature. The purpose shall be stated in terms of establishing detailed requirements, criteria, general methods, responsibilities, and overall planning to confirm, in accordance with section 4 of the MPCI Requirements Specification, that the MPCI or designated portions of it, fulfills the requirements of section 3 of the MPCI Requirements Specification. Reasons for excluding any portions of the MPCI from this test plan shall be stated.

*SECTION 2.0 REFERENCES

This section shall list all documents upon which the MPCI Test Plan is based or which relate significantly to the MPCI test effort. Documents defining the MPCI configuration to which the test plan applies shall be specifically identified.

*SECTION 3.0 TEST CONCEPTS

This section shall obtain background information required to substantiate the test philosophy and to provide information to aid in understanding and evaluating the test plan.

*SECTION 4.0 QUALIFICATION REQUIREMENTS AND CRITERIA

*4.1 This section shall contain detailed qualification requirements based on section 3 of the MPCI Requirements Specification. For ease in preparation and use, information required in this section may be published as appendixes or additional volumes of the test plan, and shall be referenced in this section.

*4.2 In addition to the detail identification of performance requirements, parameters shall be stated in terms of presence or absence of specified outputs and, wherever applicable, in terms of tolerance limits for calculated values. Methods of determining whether microcode program performance corresponds to the parameters shall be stated for each requirement. Methods may include, for example, an analysis of recorded data, examination of displays or hard-copy outputs, or equipment response to operation of the microcode program.

*4.3 Situations or conditions under which qualification must be achieved shall be specified in terms of ranges of input data values, amount of input data of required types, and specific critical values or occurrences.

*SECTION 5.0 QUALIFICATION OBJECTIVES/TEST PHASE SUMMARY

This section shall identify qualification requirements to be satisfied by (1) data from microcode programming test and evaluation, (2) preliminary qualification testing, (3) formal qualification testing, and (4) system testing. Detailed requirements shall be identified by references to section 4 above and shall be contrained in subsections shown below. Depending on the nature of the requirements, it may appear in more than one of the following subsections.

*5.1 Requirements to be Satisfied by Microcode Programming Test and Evaluation Data. This subsection shall list the qualification requirements to be satisfied by data from microcode programming test and evaluation and shall identify the data to be supplied for such qualification.

*5.2 Requirements to be Satisfied by Preliminary Qualification Testing. This subsection shall list the performance requirements against which the operation of the microcode program is to be verified during Preliminary Qualification Testing.

*5.3 Requirements to be Satisfied by Formal Qualification Testing. This subsection shall list the performance requirements to be satisfied during Formal Qualification Testing.

*5.4 Requirements to be Satisfied During System Testing. This subsection shall list the qualification requirements which, by their nature, cannot be satisfied prior to System testing and which should be included in the System Test Plan (if applicable).

*SECTION 6.0 MPCI QUALIFICATION TEST IMPLEMENTATION

This section shall contain the plan for implementation of the MPCI Preliminary and Formal Qualification Test program. Subsection 6.1 shall apply to Preliminary Qualification Tests and subsection 6.2 shall apply to Formal Qualification Tests. Each subsection shall contain:

*6.N.1 Location and Schedule. The location at which the qualification tests will be conducted shall be specified. The schedule for the tests shall be established in terms of dates for particular tests or sets of tests, general periods (weeks or months) for various tests or phases of testing, or periods relative to milestones in the overall acquisition schedule.

*6.N.2 Limitations and General Comments. General comments and limitations relative to test implementation and accomplishment of test objectives shall be included.

*6.N.3 Preparation of Inputs. General methods for preparation of input data shall be described. This shall include identification of simulation and/or data generation vehicles to be used. Requirements for review or validation of input data shall be specified.

*6.N.4 Conduct of the Tests. General procedures for test conduct and delineation of responsibilities for test direction, operation, and observation shall be established.

*6.N.5 Analysis of Results. General procedures for analysis of test results shall be described. This shall include identification of computer programs to be used for data reduction/analysis.

*6.N.6 Summarization of Equipment and Computer Program Requirements. Requirements for computer programs, microcode programs, other than the MPCI being tested, and for equipment such as logic analyzers, computers, supporting equipment, etc., shall be summarized. This information will normally correspond to requirements set forth in paragraph 4.1.1 of the MPCI Requirements Specification.

*6.N.7 Summarization of Personnel Requirements. Personnel requirements shall be summarized. This shall include statements describing responsibilities, authority, and particular knowledge of skills required. The extent of this summary shall be such that provisions for supplying personnel will be consistent with a meaningful, successful qualification effort.

*SECTION 7.0 CONTROL AND REPORTING PROCEDURES

This section shall specify requirements and procedures for controlling and documenting the MPCI Test Program. These procedures and requirements shall be specified in subsections as follows:

*7.1 Control of the MPCI Test Program. This subsection shall contain procedures for revising or updating the MPCI Test Plan as a result of schedule changes, changes to design requirements and/or MPCI detail design, revised provisions for supporting the test program, etc. Interrelation between the MPCI Test Program and control of the MPCI design requirements and/or configuration shall be established.

*7.2 Documentation of Test Procedures. This subsection shall specify requirements and procedures for preparing, reviewing, and if necessary, revising documentation of specific test procedures.

*7.3 Documentation of Test Reports. This subsection shall specify requirements and procedures for preparing and reviewing reports of individual qualification tests, summaries of the MPCI Test Program and/or its phases, and other reports which may be required related to the MPCI Test Program.

3.1.4.2 TEST PROCEDURES

Microcode test procedures are developed from test plans and microcode specification and design documents. They present detailed instructions for test setup, execution, and evaluation of test results. The procedure states in general terms the organization or structure of the test and any assumptions or constraints imposed on its usage. It describes the required equipment, manpower, microcode programs, computer programs, and supporting documentation. If various modes of operation are possible, the requirements for each are specified. Equipment required for operation is identified and revisions or modifications to these equipments are specified as well as any pretest checkout required to ensure a valid test environment.

The recommended format and content for MPCI Test Procedures document is provided:

*SECTION 1.0 CAPTION

The test procedure shall have a caption containing:

*1.1 Test Identification. The individual test shall be uniquely identified.

*1.2 Contract Item. The MPCI to which the test applies shall be identified by number and approved nomenclature.

*1.3 Primary Function. The MPCI's primary functions or segments which are to be tested shall be identified.

*SECTION 2.0 LOCATION AND SCHEDULE

The location and schedule shall be shown for the following test efforts:

a. Tests;
b. Data reduction/analysis.

*SECTION 3.0 REFERENCES

Reference documents applicable to the test shall be listed. Such references shall include:

a. MPCI Test Plan;

b. MPCI Specifications for the functions to be tested;

c. Users' manuals for the MPCI (if applicable);

d. Users' manuals for test/support microcode and/or computer programs and equipment.

*SECTION 4.0 TEST OBJECTIVES

Detailed objectives of the test shall be indicated by brief function descriptions and references to section 4 of the MPCI Test Plan.

*SECTION 5.0 MANNING AND RESPONSIBILITIES

Requirements and responsibilities shall be shown for console operators, test directors, technical consultants, data analysts, or other essential test personnel. Special knowledge or skills required shall be stated. Requirements which are identical to those stated in section 6 of the MPCI Test Plan may be specified by reference to the test plan.

*SECTION 6.0 EQUIPMENT AND COMPUTER PROGRAM REQUIREMENTS

Requirements for microcode programs, other than the MPCI being tested, and for equipment necessary to support the test shall be specified. Requirements identical to those stated in section 5 of the MPCI Test Plan may be specified by reference to the test plan.

*SECTION 7.0 TEST OPERATING PROCEDURES

Procedures for operating the microcode program to be tested shall be specified in this section. Normally such specification shall be by reference and exception to documentation listed in section 3 above. Procedures will be specified to:

*7.1 Initiate the Program or Device Operation. Procedures shall be specified to read the program into the device (if applicable), establish the required mode of operation, initially set any required parameters, provide for required inputs, and outputs, and begin operation of the program or device. Listings of input material to accomplish the above shall be provided as an appendix.

*7.2 Maintain the Program or Device Operation. Procedures shall be specified to maintain operation of the program or device whenever operator intervention is required as, for example, to maintain input data flow.

*7.3 Terminate and Restart the Program or Device Operation. Procedures shall be specified for normal and unscheduled termination of program or device operation, as well as restarting operation so as to ensure that necessary output data shall be obtained and made available for required evaluation.

*SECTION 8.0 DETAILED TEST DESCRIPTION

*8.1 This section shall describe in detail the test outputs, events, and expected results. Test objectives satisfied or partially satisfied by each expected result shall be identified by reference to section 4 of the MPCI Test Plan.

Test events shall be described in the order in which they are planned to occur with dependency of any one event on another so indicated. If more than one operating or monitoring position is involved in the test, the sequence of events for each position shall be indicated. Interdependence of operating positions with respect to specific events shall be described.

*8.2 Listings of inputs and/or listings produced in the preparation of test inputs shall be included in an appendix.

*SECTION 9.0 DATA REDUCTION AND ANALYSIS

This section shall contain the requirements and procedures for reduction and analysis of test data. The information shall be contained in subsections as follows:

*9.1 Recording and Reduction Requirements. Data which must be recorded during the test by the program, manually, and/or by instrumentation shall be specified. In addition, requirements for format and content of the data resulting from the reduction/analysis process shall be specified. Requirements for data recording and reduction shall be specified in a manner and detail such that the resulting information will clearly show whether the test objectives have been met.

*9.2 Data Reduction/Analysis Procedures.

This subsection shall contain the procedures to be employed in reducing and analyzing data resulting from the test. Normally, the procedures shall be specified by references and exception to documentation listed in section 2 above.

3.1.4.3 TEST REPORTS

MPCI test results are accumulated throughout the various levels of microcode qualification testing. On completion of the qualification test, the results are integrated into a Microcode Test Report document. The recommended format and content for an MPCI Test Report is provided:

*SECTION 1.0 MICROCODE PROGRAM CONFIGURATION ITEM IDENTIFICATION

The MPCI to which the test applies shall be identified by number and approved nomenclature.

*SECTION 2.0 TEST IDENTIFICATION

The identification of the individual qualification test as shown on the test procedure shall be shown on the test report.

*SECTION 3.0 PRIMARY FUNCTION

The MPCI's primary functions or segments to which the test applies shall be identified.

*SECTION 4.0 TEST PLAN AND PROCEDURES REFERENCE

The MPCI Test Plan and the MPCI Test Procedures for the test shall be referenced.

*SECTION 5.0 TEST RESULTS

The results of the test shall be stated as follows:

*5.1 Identification of those planned objectives for which actual test results were identical, with the expected results as specified in the test procedures, or for which variation between actual and expected results was within specified tolerances. In the latter case, actual test results shall be shown.

*5.2 Identification of those planned objectives for which actual test results differed from expected results beyond specified limits. In this case, actual test results shall be shown.

*5.3 Identification of any planned test objectives for which actual results were not obtained. Reason for not fulfilling such objectives shall be stated.

*SECTION 6.0 RECOMMENDATION

Recommendations for subsequent action shall be stated, based on the test results. Such recommendations may include:

*6.1 Revising the MPCI in order to meet specifically identified requirements which were not fulfilled.

*6.2 Revising the MPCI Requirements Specification in cases where the test results disclose ambiguity or conflicting requirements.

*6.3 Conducting additional tests to fulfill objectives for which results were not as expected.

6.4 Qualifying those functions for which test objectives have been fulfilled.

3.1.5 OPERATIONS AND MAINTENANCE

3.1.5.1 MANUALS

The Microcode User's or Operator's Manual provides the user with information that is required for the operational use of the microcoded device. The recommended format and content for User's Manual is provided:

*SECTION 1.0 SCOPE

This section shall consist of the following paragraphs:

1.1 Identification. This section shall include the configuration item index number of the system or device and the equipment to which this manual is applicable. It shall provide a statement of intended use and include:

a. Control and Implementation - List manual control implementation data and how changes may be effected.

b. Equipment and Program Description - Designate the minimal device and peripheral equipment required to initiate system operation. Any revisions or modifications to that equipment shall be noted. Equipment characteristics and capabilities and program functions, capabilities, and organization will be described.

*1.2 System Materials. This section shall describe required magnetic/paper tapes, card decks and listings (if applicable). These items shall be identified by appropriate nomenclature and configuration item number. Also, supporting documents consisting of specifications, procedures, handbooks and manuals required for system operation shall be listed.

*SECTION 2.0 APPLICABLE DOCUMENTS

*SECTION 3.0 SYSTEM OPERATION

*3.1 Power On/Off. This section shall explain the step-by-step procedures required to power-on and power-off the equipment for operational and standby mode.

*3.2 System Initiation. This section shall include instructions for preparation and set-up prior to system operation. It shall include: equipment set-up, procedures and steps required in equipment pre-operation. Explain any physical preparation such as loading, initial switch settings, etc.

*3.3 General Information. Describe procedures necessary to BOOTSTRAP the system and load programs. Explain common commands for system initiation (e.g. program interrupt/recovery and system priority organization).

3.4 Operating Procedure. This section shall describe system RESTART after system INITIATION. In the event that more than one mode of operation is available, instructions for selection of each mode will be provided. Sufficient details of all options shall be listed and instructions for recovery from error traps/halts shall be provided.

*3.5 Input/Output. Describe INPUT/OUTPUT media and explain detailed procedures required for each. List operator procedures for interactive message/replies (if applicable). Describe data insertion/modification procedures (if applicable).

*3.6 Monitor Procedures. This section shall describe the requirements for monitoring the microcode program while in operation. Trouble and malfunction indications shall be delineated, with corresponding corrective actions. Evaluation techniques for fault isolation shall be described to the maximum extent practicable. Conditions which require shutdown or aborting, and specific abort procedures shall be described. The operation shall be well defined.

*3.7 Auxiliary/Off-Line Routines. Explain detailed procedures required to operate all special routines of the device or system.

*3.8 Recovery Procedures. Explain procedures to follow for each trouble occurrence or program error. Give detailed instructions to obtain system dumps (if applicable). Describe steps to be taken by the operator to restart system operation after an abort or interruption of operation. Procedures for recording information concerning malfunction shall be included. Back-up operational procedures shall be described. Equipment available in back-up mode shall be described.

*3.9 Special Operational Procedures. Describe all special procedures related to system ALARMS, equipment failure, nonstandard operational requirements, or unique program/system management procedures.

*3.10 Program/System Control. Describe how the system will be maintained. Include procedures for enhancements, corrections, control store readouts, program patches, program change messages, and on-site support.

*3.11 Program Inventory. This paragraph shall provide an inventory of various system devices or programs.

*SECTION 4.0 DIAGNOSTIC REQUIREMENTS

*4.1 Diagnostic Data Description. This section shall contain all system data diagnostics. Each descriptor shall be listed and status values/meanings shall be described in detail.

*4.2 Diagnostic Program Description. This section shall describe each diagnostic program. It shall contain descriptions of functional tests and definitions of results. This section shall include diagnostic program schedules (if applicable) and describe system configuration during diagnostic program operation.

*4.3 External Diagnostic Programs. This section shall list external non-system diagnostic programs which are used to test system operation. A complete list of functions tested, data output and meaning will be provided.

*4.4 Special Instructions. This section shall include special instructions needed to operate diagnostic system programs.

It is understood that much of what is described in the recommended content may not be applicable to a specific microcoded device. If the microcoded device is part of a larger system such as a computer system, the portions of this manual unique to microcode should be integrated into the System or Software User's Manual.

3.1.5.2 TRAINING

Training for operation and maintenance of microcoded devices proved to be difficult with respect to establishing a standard for either training procedures or for documentation. None of the microcode developers interviewed, trained their field personnel to diagnose or modify microcode. They instead utilized their hardware training courses (if any) to allow the detection and replacement of "least replaceable units." They relied on "in-house" experts to do microcode maintenance. These personnel utilized the development documentation to understand the microcode.

4.0 CONCLUSION

Although the survey indicates a general trend toward the production of reliable, good quality microcode there are no industry standards. Procedures and documentation vary from project to project and if standards are applied, they are usually adapted from software standards. A proposed standard has been presented. In order to validate the standard and to cover unforseen difficulties, additional studies might be undertaken. These studies are:

a. A study to analyze what portions of the documentation standards proposed in this report could be combined or possibly eliminated for projects with very simple microcoding efforts. Obviously, an easily understood, vertically formatted microinstruction set using 50 to 100 instructions to perform its function can be considered simple. Is there a means by which complexity can be measured so as to specify what level of standardization is necessary for a specific task?

b. Studies to monitor contracts for which microcode is being developed. The use of standards, design and implementation tools (and techniques), and test methods could be analyzed to try to quantify such factors as reliability, maintainability, and development costs for projects which use or don't use these approaches.

c. A study to determine the advisability of using a PDL to portray a design "instead of" or "in addition to" a functional flow diagram.

d. A study to determine if common vocabulary and symbols could be developed for microassemblers. Understanding a microcoded device would then be a matter of understanding its peculiarities rather than its microcode representation.

e. A continuing study to provide periodic monitoring of Higher Level Language development and Verification techniques as they apply to microcode. These two areas are evolving and when mature enough, should become a permanent part of the microcode development process. Standards will, at that time, need to reflect their impact.

REFERENCES

1. M.V. Wilkes,"The Best Way to Design an Automatic Calculating Machine", Report of the Manchester University Computer Inaugural Conference, July 1951, pp. 16-18.

2. N.R. Firth, "The Role of Software Tools in the Development of the Eclipse MV/8000 Microcode", Proc. of 13th Microprogramming Workshop, 1980.

3. D.A. Patterson, K. Lew, R. Tuck, "Towards an Efficient, Machine-Independent Language for Microprogramming", Proceedings: MICRO 12 (November 1979), pp. 22-35.

4. J. Nash, M. Spak, "Hardware and Software Tools for the Development of a Micro-Programmed Microprocessor", Proceedings: MICRO 12 (November 1979) pp. 73-83.

5. B.A. Laws, Jr., "Microbe: A Self Commenting Microassembler", Proceedings: 10th Annual Workshop on Microprogramming, pp. 61-65.

6. W. Rottmann, "MIKADO - A System for Computer Aided Microprogram Designs", Preprints, 7th Annual Workshop on Microprogramming, pp. 195-202.

7. L.A. Belady, C.D. Evangelisti, L.R. Power, "GREENPRINT: A Graphic Representation of Structural Programs", IBM Systems Journal, Vol. 19, No. 4, 1980, pp. 542-553.

8. R.D. Nielson, "Functional Simulation of Microprogrammed LSI Circuits", Proceedings: IEEE 1978 National Aerospace and Electronics Conference (NAICON), pp. 588-593.

9. W.H. Joyner, Jr., W.C. Carter, G.B. Leeman, Jr., "Automated Proofs of Microprogram Correctness", Proceedings: 9th Annual Workshop on Microprogramming, pp. 51-55.

DA
FIL
2-